BACHELOR SOFTWARE ENGINEERING PROJECT

Unit Test Plan

Team Waterfowl

July 1, 2021

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Eindhoven University of Technology

2IPE0 Software Engineering Project
QUARTILE IV - 2020-2021

Unit Test Plan

Team Waterfowl

Authors:
Adrien Castella *(1280880)*
Adrian Cucoş *(1327860)*
Cosmin Manea *(1298542)*
Noah van der Meer *(1116703)*
Lulof Pirée *(1363638)*
Mihail Ţifrea *(1317415)*
Tristan Trouwen *(1322591)*
Tudor Voicu *(1339532)*
Adrian Vrămuleț *(1284487)*
Yuqing Zeng *(1284835)*

Supervisor:
Gerard Zwaan, univ. lecturer
Customer:
Jim Portegies, asst. prof.

Version 0.3
July 1, 2021

Abstract

This document describes the Unit Test Plan (UTP) for Waterproof, an educational environment for writing mathematical proofs in interactive notebooks. The project aims to improve the existing Waterproof software in three key areas: introducing a robust installation procedure for the Windows platform (i), designing and implementing a robust AST structure (ii), developing a new tactics library based on $L_{\text{tac}}2$ (iii). This document complies with the ESA software standards (cf. [8]).

Contents

Document Status Sheet

Title	Unit Test Plan
Authors	Adrien Castella, Adrian Cucoş, Cosmin Manea, Noah van der Meer, Lulof Pirée, Mihail Ţifrea, Tristan Trouwen, Tudor Voicu, Adrian Vrămuleț, Yuqing Zeng
Version	0.3
Status	Final
Creation Date	May 4, 2021
Last Modified	July 1, 2021

Document History log

June 29th, 2021: First version release (version 0.1).

June 30th, 2021: Second version release after an overall revision (version 0.2).

July 1st, 2021: Final revision (version 0.3)

1 Introduction

1.1 Purpose

The Unit Test Plan for the Waterfowl project specifies all of the unit tests which can be considered for testing the software developed by the Robust Waterproof project, such that all the newly added features to Waterproof work as intended and do not break compatibility with the existing software. Each of the three newly added features is composed of multiple units, grouped to structure the tests in the Unit Test Plan. This document provides enough details such that anyone can perform all of the tests.

1.2 Overview

This document consists of five chapters/sections.

In Chapter 2, the features to be tested are presented, accompanied with a short overview of the testing process.

In Chapter 3, the specifications of each test case are mentioned. Each test case will consist of a unique identifier, a short description, and the prerequisites for executing the respective test The input and output specifications are quickly mentioned for each unit test case, but they are also referred to the code tree.

In Chapter 4, the test procedures are explained. All test cases defined in Chapter 2 will be executed in corresponding logical order. Exact instructions on performing the tests will also be provided.

Finally, in Chapter 5, the results of the corresponding tests from Chapter 3 are presented, and in Chapter 6, the test coverage for each part of the Waterfowl project is shown and explained.

1.3 List of definitions

Term	Definition
Automation	A feature of Coq to automatically synthesize simple pieces of a proof.
Automatic solving	Coq commands that automatically advance the proof state.
Coq	A formal proof management system that allows for expressing mathematical expressions and assertions (cf. [4]).
CoqAST	A type of data structure returned by SerAPI which represents the code written by the user.
Coq-SerAPI	A library for machine-to-machine interaction with the Coq proof assistant.
Electron	A framework for creating cross-platform desktop applications using web technologies (JavaScript, HTML, and CSS).
Executable file	A program that can be run in the OS without explicitly needing to load it with another program.
Git	A version control system, a tool used to manage and track changes of software projects.
GitHub	A provider of Internet hosting for software development, using Git.
Lemma	Proven statement used as a stepping stone for proving a larger result.
$\mathcal{L}_{\text{tac}}1$	A meta language for Coq.
$\mathcal{L}_{\text{tac}}2$	A newer meta language for Coq (successor of $\mathcal{L}_{\text{tac}}1$).
Markdown	A format for text files that can be interpreted by a markdown viewer, but is also designed to be readable as source code. Typically uses the `.md` extension.
Mechanization	Automating a process for more efficiently accomplishing a given task.
OCaml	A general-purpose typed programming language. OCaml is designed to enhance expressiveness and safety (cf. [6]).
Opam	The OCaml Package Manager (cf. [7]). A source-based package manager for distributing OCaml programs and tools.
Proof assistants	"computer programs" specifically designed for mechanizing rigorous mathematical proofs on a computer.
Proof state	Dynamic "logbook" in Coq that shows (1) the current set of hypotheses and proven statements and (2) the statements that yet need to be proven.
README	A markdown file commonly added to the root directory of a Git repository, that provided a summary of the content of the repository.
S-expression	A symbolic notation for representing a (nested) tree-structured data.
Software dependency	A software component necessary for the operations of a different program.
Software package	A collection of applications or code modules that work together to meet various goals and objectives.

Software repository	A storage location for software packages.
Syntax highlighting	The feature displays text, especially source code, in different colours and fonts according to the category of terms.
Tactic	Mathematical statement that advances the proof state.
Tactic language	A set of tactics that together form the complete functionality of Coq.
User	Person that uses the application.
`.v` files	Coq source code files (also known as vernacular files).
Wrapper	Application built around another program in order to provide a different interface. For example, Waterproof is a wrapper around Coq that simplifies the proving language.

1.3.1 Abbreviations

API	Application Programming Interface
AST	Abstract Syntax Tree
INRIA	The French National Institute for Research in Digital Science and Technology
JS	JavaScript
SEP	Software Engineering Project
URL	Uniform Resource Locator

1.4 List of references

[1] *Auto Update - electron-builder*. URL: `https://www.electron.build/auto-update` (visited on 07/01/2021).

[2] Adrien Castella et al. *Waterfowl Acceptance Test Plan*. Eindhoven University of Technology, June 2021.

[3] Adrien Castella et al. *Waterfowl User Requirements Document*. Eindhoven University of Technology, June 2021.

[4] Thierry Coquand, Gérard Huet, and Christine Paulin. *The Coq Proof Assistant*. Accessed: 2021-04-27. URL: `https://coq.inria.fr/`.

[5] *Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS*. en-US. URL: `https://www.electronjs.org/` (visited on 05/04/2021).

[6] *OCaml*. Accessed: 2021-04-27. URL: `https://ocaml.org/index.html`.

[7] *OCaml Package Manager*. OCamlPro. URL: `https://opam.ocaml.org/` (visited on 06/26/2021).

[8] ESA Board for Software Standardisation and Control (BSSC). *ESA Software Engineering Standards: Issue 2*. 1991.

2 Test plan

2.1 Test items

The unit tests have been designed to test the software developed by team Waterfowl for the Robust Waterproof project. As described before in this document and also in the User Requirements Document (cf. [3]), the Robust Waterproof project is mainly focused on the following key aspects:

- Introducing a robust installation procedure for Waterproof and all of its dependencies for the Microsoft Windows platform, and developing an updater for Waterproof.
- Designing and implementing a robust Abstract Syntax Tree (AST) structure internally for the representation of proofs
- Developing a new tactics library based on $L_{\text{tac}}2$

All of the internals related to these three key areas will be tested.

2.2 Features to be tested

All internal procedures/methods corresponding to the three aspects listed above will be covered by unit tests. Due to the nature of the project, there are not many front-end functionalities to be tested, and the ones that do need testing are extensively covered by the Acceptance Test Plan (ATP). Therefore, no unit tests have been designed for front-end features. No unit tests were designed for the existing Waterproof software, as this is considered to be a mature project that has been properly tested in the past. For more information, refer to the Acceptance Test Plan (cf. [2]) and the User Requirements Document ([3]) of the Waterfowl project.

2.3 Test deliverables

The following items will be delivered before testing starts:

- Chapters 1 through 4 of the Unit Test Plan document.
- The User Requirements Document.
- The software developed within the Waterfowl project.

After the tests have been completed successfully, the following items will be delivered:

- The complete, finalised Unit Test Plan document.
- Problem reports (if any).

2.4 Testing tasks

Firstly, all environmental needs for this part (cf. Section 2.5) must be ensured before unit testing the installer. One of the main differences is that the Waterfowl installer was specifically designed for the Windows 10 operating system.

2.4.1 Waterfowl Installer

The Waterfowl installer is based around the Coq platform which itself is thoroughly tested. The Coq platform creates installer configuration files in accordance with the NSIS framework and builds an installer based on these files. The installer code for the Waterproof dependency installer extends this by using a configuration file that specifies which libraries to be included in the installer.

The unit tests that will be performed essentially test whether or not opam and custom packages in the specified configuration file are also included in the installer creation configuration files. It will also test whether the default selection option from the configuration is incorporated in the installer creation configuration files.

The updater component of the Waterproof application is based on the Electron-Updater library (cf. [5]) and is integrated into the existing Waterproof code structure, following the best practices suggested by the official Electron-Updater library [1]. Moreover, following this structure, the updater does not have functions or classes that can be unit tested. The reader is referred to the Waterfowl Acceptance Test Plan (cf. [2]) for the functional tests for the updater.

We recall that the Waterfowl installer was designed for the Windows 10 operating system. Therefore, before performing the unit tests for the Abstract Syntax Tree development part and for the New Tactics Library development part, the following tasks need to be executed:

- Switch to a Linux OS environment that supports the Coq proof assistant (such as Ubuntu 20.04), as specified in Section 2.5.
- Ensure that all environmental needs (cf. Section 2.5) are met.

Let us now describe the testing tasks for the remaining two parts of the Waterfowl project, starting with the AST parser.

2.4.2 AST parser - parsing correctness

The first testing task of the AST parser consists of checking the parsing correctness. To be more specific, it tests if the AST-Parser creates the correct internal data structure for a valid S-expression, which is the essential function the of AST parser (c.f URD requirement `CAR-AST-1`).

2.4.3 AST parser - robustness given unexpected behaviour

The parser should function correctly for well-formatted input. However, for input with syntax errors or unexpected types, the parser should ideally isolate the problem and throw an informative error. In this section, we test for each object type that the parser can detect wrong format and throw an error.

2.4.4 AST parser - the flatten operation

AST flattening is an operation that takes a parsed AST object hierarchy and returns a mapping between object type and location. This step tests if the parser throws errors on empty-ASTs and if the parser returns all relevant object types with the correct corresponding location.

2.4.5 AST parser - the pretty print operation

A feature that is important for the developers of waterproof is the ability to visually see the output of the parser. This allows for easy validation of the correctness of the result, and it is also a useful feature when adding support for new types or features relying on the AST-parser.

This step tests if the parser will properly pretty-print the AST for a given class (with possible sub-classes) and if the shared function which handles functionality of the pretty-print works correctly.

2.4.6 AST parser - run-time performance

The given step tests the AST parser's performance. More specifically, it tests whether the parser can handle large input (over 10000 characters) in a reasonable amount of time (less than 0.4 seconds) (cf URD requirement `COR-AST-1`).

Lastly, the testing tasks for the new tactics library are the following.

2.4.7 New tactics library - behaviour of each tactic

We remark that Coq is a proof assistant and $L_{\text{tac}}2$ is a **functional programming language**. This means that object-oriented programming concepts cannot be applied within $L_{\text{tac}}2$ (there are no classes etc.); everything is written in terms of functions.

As previously mentioned, one of the goals of the Waterfowl project is to create a new tactics library, written in $L_{\text{tac}}2$, and each tactic that we have written is just a function.

The first testing task is about the behaviour of each tactic (i.e. of each function written). More precisely, we will perform unit tests for the functionality of each written tactic.

2.4.8 New tactics library - configurability of the hints databases

The second testing task is about the configurability of the hints databases.

Some tactics that were written use the concept of `automation`. This `automation` is done through `hints databases`, which are collections of lemmas (either proven by us, or proven in a different Coq library). Remark that these hints databases are not databases per se, but but just a collection of proven lemmas.

The `automation` in Coq is usually done via the `(e)auto` tactic, which can take as input various hints databases. However, we have designed a customizable tactic that performs this `automation`.

The unit tests will thus be designed in order to test the functionality of our tactic that performs automation. We will also test the configurability of the hints databases for this particular tactic.

2.5 Environmental needs

As explained before, the Waterfowl project is aimed at improving three key aspects of the Waterproof software. Due to the nature of the first aspect (the installation procedure), we will distinguish between the environments used for testing this first aspect, and the other two aspects. The User Requirements Document (URD) contains a clear specification of the features and requirements related to each of these aspects.

Testing the installer is done by running a GitHub action. Hence the process is similar to how the installer is compiled and only requires a web browser.

Testing the features and requirements of the AST parser requires an environment similar with the one used for developing features on the original Waterproof software.

- While the project itself is OS-agnostic, we recommend that the testing machine runs a modern version of a Linux operating system, which support the Coq proof assistant (such as Ubuntu 20.04). Any other operating system which support Coq can be in theory used, but those setups are not directly supported.
- The development utilities used in the creation of Waterproof. This include the Node.js runtime and the project specific dependencies of Waterproof, which can be installed using the command:

```
npm install
```

For testing features and requirements related to the development of the new tactics library, the following resources will be needed:

- A desktop or laptop computer running a version of the Linux operating system which supports the Coq proof assistant (such as Ubuntu 20.04).
- The Coq proof assistant, which is a back-end dependency of the Waterproof software.

2.6 Test case pass/fail criteria

In Chapter 3, the unit test cases are specified in detail. A test case is passed if the received feedback matches the output specified for the test case. If the input matches the specification, but the provided feedback differs in any way from the expected output, the test case fails. If a test case fails, the overall Unit Test Plan is considered to have failed. If all tests pass, then the overall Unit Test Plan is considered passed.

3 Test case specifications

As described in Section 2.1, the Waterfowl project is divided into three parts. This chapter will hence be split into three parts: unit tests for the installer development part of the project, unit tests for the Abstract Syntax Tree development part of the project and unit tests for the new tactics library development part of the project.

3.1 Unit Test Plan for the Installer development part of the project

All of the unit tests mentioned in this section require that all changes are pushed to the GitHub repository

`https://github.com/impermeable/waterproof-dependencies-installer`

Furthermore, the user needs to have access to a browser with an internet connection to see the results.

3.1.1 `install_custom_package`

Identifier:	**UTP-1**
Testing subject:	The code in ‘install_packages.sh’ and ‘install_custom_nsis.sh’.
I/O’s:	Run the GitHub workflow Windows Tests.

3.1.2 `install_package`

Identifier:	**UTP-2**
Testing subject:	The code in ‘install_packages.sh’.
I/O’s:	Run the GitHub workflow Windows Tests.

3.1.3 `unselect_package`

Identifier:	**UTP-3**
Testing subject:	The code in ‘install_packages.sh’ and ‘unselect_packages.sh’.
I/O’s:	Run the GitHub workflow Windows Tests.

3.2 Unit Test Plan for the Abstract Syntax Tree development part of the project

The following tests cover the AST-parser component of our project. They are organised according to the structure mentioned in Sections 2.4.2 to 2.4.6.

The test cases are located inside the Waterproof project, inside the

`tests/unit/serapi/coqast`

folder.

In the context of the following tests, when we refer to all the files which contain classes subtyping CoqType, we refer to the files inside the folder `src/coq/serapi/datastructures`.

The prerequisite for **all** the following tests is that the user has cloned the Waterproof repository and they have successfully installed all dependencies required as per Section 2.5. This prerequisite will not be repeated for each test in this section.

Parsing correctness

3.2.1 `empty_s-expression`

Identifier:	**UTP-4**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command 'npm run test-ast

3.2.2 `empty_CoqAst_s-expression`

Identifier:	**UTP-5**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.2.3 `simple_s-expression`

Identifier:	**UTP-6**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.2.4 `require_import_s-expression`

Identifier:	**UTP-7**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.3 `hints_s-expression`

Identifier:	**UTP-8**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.3.1 `Complete_`L_{tac}`2_s-expression`

Identifier:	**UTP-9**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

Robustness given unexpected behaviour

3.3.2 `wrong_input_returns_error`

Identifier:	**UTP-10**
Testing subject:	CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.3.3 `correct_input_returns_object`

Identifier:	**UTP-11**
Testing subject:	CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

Pretty print operation

3.3.4 `sprintf_works_correctly`

The pretty print functionality relies on a method called `sprintf`. This method takes a string with placeholders in the form of `%s` and 0 or more additional arguments and proceeds to replace the placeholders by mapping the n-th argument with the n-th placeholder, if it exists.

Identifier:	**UTP-12**
Testing subject:	The `sprintf` function located in the file CoqType.ts
I/O's:	run the command `npm run test-ast`

3.3.5 `cprint_works_correctly`

It also takes advantage of a method called `cprint` which, given a `CoqType` subclass, prints its contents.

Identifier:	**UTP-13**
Testing subject:	The `cprint` function located in the file CoqType.ts
I/O's:	run the command `npm run test-ast`

3.3.6 `pprint_every_CoqType`

Identifier:	**UTP-14**
Testing subject:	CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

Flatten operation

3.3.7 `flatten_empty_s-expression`

Identifier:	**UTP-15**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.3.8 `flatten_every_CoqType`

Identifier:	**UTP-16**
Testing subject:	CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

Run-time_performance

3.3.9 `large_s-expression`

Identifier:	**UTP-17**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.3.10 `large_notebook`

Identifier:	**UTP-18**
Testing subject:	The function `convertToASTComp`, CoqType.ts and all the files containing classes subtyping from CoqType.
I/O's:	run the command `npm run test-ast`

3.4 Unit Test Plan for the New Tactics library development part of the project

The following tests will now be about the development of the new tactics library (cf.2.1). The prerequisite for **all** the coming unit tests is that the user has cloned the Git repository of the GitHub repository

`https://github.com/impermeable/coq-waterproof`

and saved it on their local machine, in a particular folder (this will also be mentioned in Section 4; for this reason, we will not write the prerequisite for each test again). Let us call this folder `branch`.

Moreover, all the unit testing files are located in a folder path starting with

`branch/waterproof/test/`

3.4.1 `test_load_database.v`

Identifier:	**UTP-19**
Testing subject:	The functions needed for configurability of the hints databases: `databases.v`, `selected_databases`, `AbsoluteValue.v`, `Additional.v`, `All.v`, `Exponential.v`, `Multiplication.v`, `Other.v`, `PlusMinus.v`, `RealsAndIntegers.v`, `Sets.v`, `SquareRoot.v`, `ZeroOne.v`, `reals.v` from the folder `load_database`, but also the `waterprove` automation function, from the `waterprove.v` file. in the `waterprove` folder.
I/O's:	Run the `test_load_database.v` file.

3.4.2 `test_set_intuition.v`

Identifier:	**UTP-20**
Testing subject:	The functions from the files `Enabled.v` and `Disabled.v` in the folder `set_intuition`, but also the `waterprove.v` file in the `waterprove` folder.
I/O's:	Run the `test_set_intuition.v` file.

3.4.3 `test_set_search_depth.v`

Identifier:	**UTP-21**
Testing subject:	The functions from the files `To_1.v`, `To_2.v`, `To_3.v`, `To_4.v` and `To_5.v` from the folder `set_search_depth`, but also the `waterprove.v` file in the `waterprove` folder.
I/O's:	Run the `test_set_search_depth.v` file.

3.4.4 `test_auxiliary_test.v`

Identifier:	**UTP-22**
Testing subject:	The functions from the auxiliary file `test_auxiliary.v`.
I/O's:	Run the `test_auxiliary_test.v` file.

3.4.5 `string_auxiliary_test.v`

Identifier:	**UTP-23**
Testing subject:	The functions from the auxiliary file `string_auxiliary.v`.
I/O's:	Run the `string_auxiliary_test.v` file.

3.4.6 `test_basic_contradiction.v`

Identifier:	**UTP-24**
Testing subject:	The functions and tactics from the file `basic_contradiction`, in the folder `contradiction_tactics`.
I/O's:	Run the `test_basic_contradiction.v` file in the `test_contradiction_tactics` folder.

3.4.7 `assume_test.v`

Identifier:	**UTP-25**
Testing subject:	The functions, together with the `Assume` tactic in the file `assume.v`.
I/O's:	Run the `assume_test.v` file in the `test_tactics` folder.

3.4.8 `goal_to_hint_test.v`

Identifier:	**UTP-26**
Testing subject:	The functions, together with the `Help` tactic in the file `goal_to_hint.v`.
I/O's:	Run the `goal_to_hint_test.v` file in the `test_tactics` folder.

3.4.9 `rewrite_inequalities_test.v`

Identifier:	**UTP-27**
Testing subject:	The functions, together with the `Rewrite equality` and `Rewrite inequality` tactics in the file `rewrite_inequalities.v`.
I/O's:	Run the `rewrite_inequalities_test.v` file in the `test_tactics` folder.

3.4.10 `such_that_test.v`

Identifier:	**UTP-28**
Testing subject:	The functions, together with the `such that` tactic in the file `assume.v`.
I/O's:	Run the `such_that_test.v` file in the `test_tactics` folder.

3.4.11 `take_test.v`

Identifier:	**UTP-29**
Testing subject:	The functions, together with the `Take` tactic in the file `take.v`.
I/O's:	Run the `take_test.v` file in the `test_tactics` folder.

3.4.12 `test_because.v`

Identifier:	**UTP-30**
Testing subject:	The functions, together with the `Because` tactic in the file `because.v`.
I/O's:	Run the `test_because.v` file in the `test_tactics` folder.

3.4.13 `test_choose.v`

Identifier:	**UTP-31**
Testing subject:	The functions, together with the `Choose` tactic in the file `choose.v`.
I/O's:	Run the `test_choose.v` file in the `test_tactics` folder.

3.4.14 `test_choose_such_that.v`

Identifier:	**UTP-32**
Testing subject:	The functions, together with the `Choose such that` tactic in the file `choose_such_that.v`.
I/O's:	Run the `test_choose_such_that.v` file in the `test_tactics` folder.

3.4.15 `test_either.v`

Identifier:	**UTP-33**
Testing subject:	The functions, together with the `Either` tactic in the file `either.v` and its auxiliary lemmas in in the `decidability_db.v` file.
I/O's:	Run the `test_either.v` file in the `test_tactics` folder.

3.4.16 `test_we_know.v`

Identifier:	**UTP-34**
Testing subject:	The functions, together with the `We know` tactic in the file `we_know.v`.
I/O's:	Run the `test_we_know.v` file in the `test_tactics` folder.

3.4.17 `test_we_show_both_statements.v`

Identifier:	**UTP-35**
Testing subject:	The functions, together with the `We show both statements` tactic in the file `we_show_both_statements.v`.
I/O's:	Run the `test_we_show_both_statements.v` file in the `test_tactics` folder.

3.4.18 `test_we_show_both_directions.v`

Identifier:	**UTP-36**
Testing subject:	The functions, together with the `We show both directions` tactic in the file `we_show_both_directions.v`.
I/O's:	Run the `test_we_show_both_directions.v` file in the `test_tactics` folder.

3.4.19 `unfold_test.v`

Identifier:	**UTP-37**
Testing subject:	The functions, together with the `Unfold` and `Expand the definition of` tactics in the file `unfold.v`.
I/O's:	Run the `unfold_test.v` file in the `test_tactics` folder.

3.4.20 `we_need_to_show_test.v`

Identifier:	**UTP-38**
Testing subject:	The functions, together with the `We need to show` tactic and its variations in the file `we_need_to_show.v`.
I/O's:	Run the `we_need_to_show_test.v` file in the `test_tactics` folder.

3.4.21 `test_sets_automation_tactic.v`

Identifier:	**UTP-39**
Testing subject:	The functions, together with the `This set equality is trivial` and `We prove equality by proving two set inclustions` tactics in the file `sets_automation_tactics.v`.
I/O's:	Run the `test_sets_automation_tactic.v` file in the `test_tactics/test_sets_tactics` folder.

3.4.22 `it_holds_that_test.v`

Identifier:	**UTP-40**
Testing subject:	The functions, together with the `It holds that` tactic and its variations, in the file `it_holds_that.v`.
I/O's:	Run the `it_holds_that_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.23 `it_suffices_to_show_test.v`

Identifier:	**UTP-41**
Testing subject:	The functions, together with the `It suffices to show` tactic and its variations, in the file `it_suffices_to_show.v`.
I/O's:	Run the `it_suffices_to_show_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.24 `proof_finishing_tactics_test.v`

Identifier:	**UTP-42**
Testing subject:	The functions, together with the proof finishing tactics in the file `proof_finishing_tactics.v`.
I/O's:	Run the `proof_finishing_tactics_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.25 `rewrite_using_test.v`

Identifier:	**UTP-43**
Testing subject:	The functions, together with the `Rewrite using` tactic and its variations, in the file `rewrite_using.v`.
I/O's:	Run the `rewrite_using_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.26 `test_apply.v`

Identifier:	**UTP-44**
Testing subject:	The functions, together with the `Apply` tactic in the file `apply.v`.
I/O's:	Run the `test_apply.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.27 `test_claims.v`

Identifier:	**UTP-45**
Testing subject:	The functions, together with the `We claim` tactic in the file `claims.v`.
I/O's:	Run the `test_claims.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.28 `test_define.v`

Identifier:	**UTP-46**
Testing subject:	The functions, together with the `Define` tactic in the file `define.v`.
I/O's:	Run the `test_define.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.29 `test_induction.v`

Identifier:	**UTP-47**
Testing subject:	The functions, together with the `We prove by induction on` tactic in the file `induction.v`.
I/O's:	Run the `test_induction.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.30 `test_simplify.v`

Identifier:	**UTP-48**
Testing subject:	The functions, together with the `Simplify what we need to show` tactic in the file `induction.v`.
I/O's:	Run the `test_simplify.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.31 `we_conclude_that_test.v`

Identifier:	**UTP-49**
Testing subject:	The functions, together with the `We conclude that` tactic and its variations in the file `we_conclude_that.v`.
I/O's:	Run the `we_conclude_that_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.32 `write_as_test.v`

Identifier:	**UTP-50**
Testing subject:	The functions, together with the `Write as` tactic and its variations in the file `write_as.v`.
I/O's:	Run the `write_as_test.v` file in the `test_tactics/test_forward_reasoning` folder.

3.4.33 `write_using_test.v`

Identifier:	**UTP-51**
Testing subject:	The functions, together with the `Write using` tactic and its variations in the file `write_using.v`.
I/O's:	Run the `write_using_test.v` file in the `test_tactics/test_forward_reasoning` folder.

4 Test procedures

4.1 Unit test procedures Installer

4.1.1 Purpose

The procedure aims to run all tests for the dependencies installer of the Waterfowl project. Failure of test 3.1.1 indicates custom packages from GitHub are not installed incorrectly. Failure of test 3.1.2 indicates that opam packages are not installed incorrectly. Failure of test 3.1.3 indicates that the selection settings for the executable installer's 'Choose components' window are not carried out correctly.

4.1.2 Procedure steps

Open the software's Github Actions page at

`https://github.com/impermeable/waterproof-dependencies-installer/actions/`.

The 'Windows tests' action will be automatically executed on each new commit. A possible test can be stopped by pressing the dots next to the workflow and pressing 'cancel run'. The log can be shown by clicking the workflow and clicking again on one of the jobs.

The workflow will leave a green checkmark when all tests were successful; a cross will be shown if not. In the latter case, one can open the workflow report and open the 'Run test file' part to show the test file's output indicating which test failed and which test passed.

Code coverage is measured manually by investigating the shell code executed. Notice that the shell code runs sequentially and contains minimal logic. Hence, it is easy for the code coverage by unit tests to be 100% since each file has to be run top to bottom to function properly.

4.2 Unit test procedures Abstract Syntax Tree

4.2.1 Purpose

This procedure aims to run all tests as implemented for the Abstract Syntax Tree Parser of the Waterfowl project. Failure of tests from 3.2.1-4 indicates a malfunction in Parsing. Failure of tests in 3.2.5-6 indicates a problem with robustness and error-throwing. 3.27-8 indicates problems with flattening ASTs and possible defects in syntax highlighting. Failure of tests in 3.2.8-9 indicates the parser has not reached the desired performance requirements. Thus if all test cases pass, one has confidence the parser should conform to the expected behaviours.

4.2.2 Procedure steps

The GitHub repository

`https://github.com/impermeable/coq-waterproof`

must be cloned and saved on the local machine, in a particular folder. Let us call this folder `branch`.

We assume that the environment requirements specified in Section 2.5 are satisfied

The testing procedure is as follows:

1. Open the Linux terminal and go to the folder `branch`.
2. Run the command `npm run test-ast`.
3. Wait for this operation to finish successfully, without errors.
4. Upon a successful run, a list of all the executed tests will be shown, along with their results. In the end, the following message will be shown

```
MOCHA Tests completed successfully
```

The test runner provides information about the total execution time of the tests. Tests can be restarted by executing the same command mentioned above.

As the testing environment runs inside a UNIX terminal shell, measurements and cancellation are done as with any other Linux terminal program, via POSIX-compliant commands.

Code coverage can also be measured automatically by using the test runner.

4.3 Unit test procedures New tactics library

4.3.1 Purpose

The purpose of this procedure it to run all tests as implemented in the New Tactics Library of the Waterfowl project. If one or more tests fail, this might indicate a malfunctioning part of the tactics library.

4.3.2 Procedure steps

To be able to run the tests, a version of the Linux operating system that supports the Coq proof assistant should be used, as explained in Section 2.5 (such as Ubuntu 20.04). Moreover, the Coq proof assistant should also be installed on the respective system.

Finally, the GitHub repository

https://github.com/impermeable/coq-waterproof

must be cloned and saved on the local machine, in a particular folder. Let us call this folder `branch`.

Now, let us present the testing procedure:

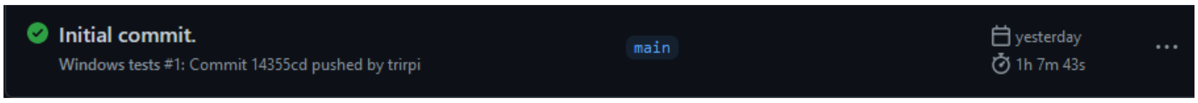
1. Open the Linux terminal and go to the folder `branch`.
2. Run the command `make`.
3. Wait for this operation to finish successfully, without errors.

The `make` operation compiles the entire tactics library, together with the respective test cases mentioned in Section 3.4. If the operation finishes completely, without any errors, it means that all the unit test cases passed.

5 Test reports

5.1 Installer

In Figure 1, the installer tests success is shown.



Initial commit.
Windows tests #1: Commit 14355cd pushed by tripi
main
yesterday
1h 7m 43s

Figure 1: Test output installer

In addition, more detailed feedback is shown in the following output from the "Run test file" workflow part.

```
[*] Test 1: Opam installation success!
[*] Test 2: Custom package installation success!
[*] Test 3: Unselection success!
```

5.2 Abstract Syntax Tree

After executing the test procedure, feedback for each unit test and the total runtime is shown as follows:

```
> waterproof@0.5.3-sep-beta test-ast
> vue-cli-service test:unit "tests/unit/serapi/coqast/**/*.spec.(js|ts)"

 WEBPACK  Compiling...

Starting type checking service...
Using 1 worker with 2048MB memory limit
 DONE  Compiled successfully in 9428ms

 WEBPACK  Compiled successfully in 9428ms

 MOCHA  Testing...


  Error checking for CoqAST objects
    Tests for the type CApp:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CLambdaN:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CLocalAssum:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CNotation:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CoqAst:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CPrim:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CProdN:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type CRef:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type DefineBody:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type GenericVType:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type HintsReference:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type HintsResolve:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type IDt:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type InConstrEntry:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type IntroIdentifier:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type IntroNaming:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type KerName:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type Ser_Qualid:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacAlias:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacApply:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacArg:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacAtom:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacCall:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacFun:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacIntroPattern:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacReduce:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacRewrite:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacThen:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type TacticDefinition:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type v:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacAssumption:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacOpenCloseScope:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacDefinition:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacEndProof:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacExpr:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacExtend:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacHints:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacProof:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacRequire:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
    Tests for the type VernacStartTheoremProof:
      ✓ should throw an error on wrong input types
      ✓ should create a class on good input
```

```
AST flattening
  ✓ should throw error on empty ast
  types with locinfo should return their location
    ✓ CoqAst should return its location
    ✓ CApp should return its location
    ✓ CLambdaN should return its location
    ✓ CLocalAssum should return its location
    ✓ CProdN should return its location
    ✓ CRef should return its location
    ✓ HintsReference should return its location
    ✓ VernacDefinition should return its location
    ✓ DefineBody should return its location
    ✓ TacCall should return its location
    ✓ VernacStartTheoremProof should return its location
    ✓ TacAlias should return its location
    ✓ TacApply should return its location
    ✓ TacAtom should return its location
    ✓ TacticDefinition should return its location
  types without locinfo should return empty
    ✓ CNotation should return no location
    ✓ CPrim should return no location
    ✓ GenericVType should return no location
    ✓ HintsResolve should return no location
    ✓ IDt should return no location
    ✓ InConstrEntry should return no location
    ✓ IntroIdentifier should return no location
    ✓ IntroNaming should return no location
    ✓ KerName should return no location
    ✓ Ser_Qualid should return no location
    ✓ TacArg should return no location
    ✓ TacFun should return no location
    ✓ TacIntroPattern should return no location
    ✓ TacReduce should return no location
    ✓ TacRewrite should return no location
    ✓ TacThen should return no location
    ✓ v should return no location
    ✓ VernacAssumption should return no location
    ✓ VernacOpenCloseScope should return no location
    ✓ VernacEndProof should return no location
    ✓ VernacExpr should return no location
    ✓ VernacExtend should return no location
    ✓ VernacHints should return no location
    ✓ VernacProof should return no location
    ✓ VernacRequire should return no location

Parsing efficiency
  ✓ should parse a long SExpr (17566 chars) quickly
  ✓ should parse many SExprs quickly (5744 chars)

Parsing CoqASTs
  ✓ should parse an empty Coq AST s-expr correctly
  ✓ shoud produce empty AST for empty S-Expr
  ✓ should parse simple S-Expr
  ✓ should parse Require Import sexpr
  ✓ should parse Hint Sexpr
  ✓ should parse a complete Ltac proof

Pretty-printer
  helper functions
    ✓ sprinf with 0 inputs
    ✓ sprinf with 2 inputs
    ✓ sprintf with n inputs
    ✓ cprint with array
    ✓ cprint with object
    ✓ should pprint CApp correctly
    ✓ should pprint CLambdaN correctly
    ✓ should pprint CLocalAssum correctly
    ✓ should pprint CNotation correctly
    ✓ should pprint CoqAst correctly
    ✓ should pprint CPrim correctly
    ✓ should pprint CProdN correctly
    ✓ should pprint CRef correctly
    ✓ should pprint DefineBody correctly
    ✓ should pprint GenericVType correctly
    ✓ should pprint HintsReference correctly
    ✓ should pprint HintsResolve correctly
    ✓ should pprint IDt correctly
    ✓ should pprint InConstrEntry correctly
    ✓ should pprint IntroIdentifier correctly
    ✓ should pprint IntroNaming correctly
    ✓ should pprint KerName correctly
    ✓ should pprint Ser_Qualid correctly
    ✓ should pprint TacAlias correctly
    ✓ should pprint TacApply correctly
```

```
      ✓ should pprint TacArg correctly
      ✓ should pprint TacAtom correctly
      ✓ should pprint TacCall correctly
      ✓ should pprint TacFun correctly
      ✓ should pprint TacIntroPattern correctly
      ✓ should pprint TacReduce correctly
      ✓ should pprint TacRewrite correctly
      ✓ should pprint TacThen correctly
      ✓ should pprint TacticDefinition correctly
      ✓ should pprint v correctly
      ✓ should pprint VernacAssumption correctly
      ✓ should pprint VernacOpenCloseScope correctly
      ✓ should pprint VernacDefinition correctly
      ✓ should pprint VernacEndProof correctly
      ✓ should pprint VernacExpr correctly
      ✓ should pprint VernacExtend correctly
      ✓ should pprint VernacHints correctly
      ✓ should pprint VernacProof correctly
      ✓ should pprint VernacRequire correctly
      ✓ should pprint VernacStartTheoremProof correctly


  174 passing (119ms)

 MOCHA  Tests completed successfully
```

5.3 New Tactics Library

After finishing the procedure displayed in Section 4.3, the following is shown in the Linux terminal:

```
coq_makefile -f _CoqProject -o CoqMakefile
make --no-print-directory -f CoqMakefile
COQDEP VFILES
COQC waterproof/auxiliary.v
COQC waterproof/definitions/set_definitions.v
COQC waterproof/notations/notations.v
File "./waterproof/notations/notations.v", line 108, characters 0-40:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/notations/notations.v", line 109, characters 0-39:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/notations/notations.v", line 110, characters 0-39:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/notations/notations.v", line 111, characters 0-38:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/notations/notations.v", line 112, characters 0-38:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/notations/notations.v", line 132, characters 0-69:
Warning: Declaring a scope implicitly is deprecated; use in advance an
explicit "Declare Scope metric_scope.". [undeclared-scope,deprecated]
File "./waterproof/notations/notations.v", line 141, characters 0-93:
Warning: Declaring a scope implicitly is deprecated; use in advance an
explicit "Declare Scope extra.". [undeclared-scope,deprecated]
File "./waterproof/notations/notations.v", line 169, characters 0-68:
Warning: Declaring a scope implicitly is deprecated; use in advance an
explicit "Declare Scope analysis_scope.". [undeclared-scope,deprecated]
COQC waterproof/databases.v
File "./waterproof/databases.v", line 108, characters 0-37:
Warning: Declaring arbitrary terms as hints is fragile; it is recommended to
declare a toplevel constant instead [fragile-hint-constr,automation]
File "./waterproof/databases.v", line 312, characters 0-43:
Warning: The default value for hint locality is currently "local" in a
```

```
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 334, characters 0-116:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 339, characters 0-57:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 353, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 354, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 358, characters 0-50:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 359, characters 0-50:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 363, characters 0-54:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 365, characters 0-56:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 367, characters 0-65:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 369, characters 0-65:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 375, characters 0-44:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 380, characters 0-54:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 382, characters 0-55:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 384, characters 0-55:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
```

```
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 386, characters 0-55:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 388, characters 0-63:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 390, characters 0-63:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 392, characters 0-53:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 397, characters 0-53:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 398, characters 0-51:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 399, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 400, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 401, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 408, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 409, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 410, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 411, characters 0-49:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 412, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
```

```
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 413, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 414, characters 0-44:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 417, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 418, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 419, characters 0-44:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 420, characters 0-43:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 421, characters 0-52:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 433, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 435, characters 0-51:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 437, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 441, characters 0-52:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 443, characters 0-50:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 445, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 447, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
```

```
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 449, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 451, characters 0-51:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 453, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 463, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 464, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 465, characters 0-53:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 466, characters 0-48:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 467, characters 0-53:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 468, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 469, characters 0-52:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 470, characters 0-47:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 478, characters 0-44:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 479, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/databases.v", line 480, characters 0-46:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
```

```
File "./waterproof/databases.v", line 481, characters 0-44:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
COQC waterproof/string_auxiliary.v
COQC waterproof/notations/set_notations.v
File "./waterproof/notations/set_notations.v", line 51, characters 0-67:
Warning: Declaring a scope implicitly is deprecated; use in advance an
explicit "Declare Scope ensemble_scope.". [undeclared-scope,deprecated]
COQC waterproof/test_auxiliary.v
COQC waterproof/selected_databases.v
COQC waterproof/waterprove/waterprove.v
COQC waterproof/load_database/AbsoluteValue.v
COQC waterproof/load_database/Exponential.v
COQC waterproof/load_database/Multiplication.v
COQC waterproof/load_database/Other.v
COQC waterproof/load_database/PlusMinus.v
COQC waterproof/load_database/SquareRoot.v
COQC waterproof/load_database/ZeroOne.v
COQC waterproof/load_database/RealsAndIntegers.v
COQC waterproof/load_database/Sets.v
COQC waterproof/load_database/Additional.v
COQC waterproof/load_database/reals.v
COQC waterproof/load_database/All.v
COQC waterproof/set_search_depth/To_1.v
COQC waterproof/set_search_depth/To_2.v
COQC waterproof/set_search_depth/To_3.v
COQC waterproof/set_search_depth/To_4.v
COQC waterproof/set_search_depth/To_5.v
COQC waterproof/set_intuition/Enabled.v
COQC waterproof/set_intuition/Disabled.v
COQC waterproof/load_database/DisableWildcard.v
COQC waterproof/load_database/EnableWildcard.v
COQC waterproof/tactics/take.v
COQC waterproof/tactics/assume.v
File "./waterproof/tactics/assume.v", line 226, characters 4-26:
Warning: The following expression should have type unit. [not-unit,ltac]
File "./waterproof/tactics/assume.v", line 238, characters 4-31:
Warning: The following expression should have type unit. [not-unit,ltac]
COQC waterproof/tactics/automation_databases/decidability_db.v
COQC waterproof/tactics/because.v
COQC waterproof/tactics/choose.v
COQC waterproof/tactics/choose_such_that.v
COQC waterproof/tactics/either.v
COQC waterproof/tactics/unfold.v
COQC waterproof/tactics/we_know.v
COQC waterproof/tactics/we_need_to_show.v
COQC waterproof/tactics/we_show_both_directions.v
COQC waterproof/tactics/we_show_both_statements.v
COQC waterproof/tactics/goal_to_hint.v
COQC waterproof/tactics/forward_reasoning/forward_reasoning_aux.v
COQC waterproof/tactics/forward_reasoning/rewrite_using.v
COQC waterproof/tactics/rewrite_inequalities.v
COQC waterproof/contradiction_tactics/basic_contradiction.v
COQC waterproof/tactics/forward_reasoning/apply.v
COQC waterproof/tactics/forward_reasoning/claims.v
COQC waterproof/tactics/forward_reasoning/define.v
COQC waterproof/tactics/forward_reasoning/induction.v
COQC waterproof/tactics/forward_reasoning/it_holds_that.v
COQC waterproof/tactics/forward_reasoning/it_suffices_to_show.v
COQC waterproof/tactics/forward_reasoning/we_conclude_that.v
COQC waterproof/tactics/forward_reasoning/proof_finishing_tactics.v
COQC waterproof/tactics/forward_reasoning/simplify.v
COQC waterproof/tactics/forward_reasoning/write_using.v
COQC waterproof/tactics/forward_reasoning/write_as.v
COQC waterproof/tactics/sets_tactics/sets_automation_tactics.v
File "./waterproof/tactics/sets_tactics/sets_automation_tactics.v", line 36, characters 0-62:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/tactics/sets_tactics/sets_automation_tactics.v", line 104, characters
    0-70:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
COQC waterproof/AllTactics.v
COQC waterproof/definitions/subsequences_definitions.v
File "./waterproof/definitions/subsequences_definitions.v", line 45, characters 0-26:
Warning: Interpreting this declaration as if a global declaration prefixed by
```

```
"Local", i.e. as a global declaration which shall not be available without
qualification when imported. [local-declaration,scope]
The goal has a forall-quantifier  ().
You may need to introduce an arbitrary variable of type  .
This can for example be done using 'Take ... : ...'.
Or you may need to make an assumption stating  .
This can for example be done using 'Assume ... : ...'.
family
     : Type
Inductive Coq.Reals.Rtopology.family

     : Set
true
Hypotheses successfully assumed
true
Hypotheses successfully assumed
Successfully rewrote goal using '(k = 0%nat)'.
The goal is indeed: (g 0   g 0)%nat
No hint available.
true
Hypotheses successfully assumed
No hint available.
New sublemma successfully added.
No hint available.
New sublemma successfully added.
No hint available.
New sublemma successfully added.
Successfully rewrote goal using '(k = S 1)'.
No hint available.
New sublemma successfully added.
true
Hypotheses successfully assumed
true
Hypotheses successfully assumed
The goal is indeed: (n k1   n k2)%nat
No hint available.
New sublemma successfully added.
No hint available.
New sublemma successfully added.
Successfully rewrote goal using '(S k = (k + 1)%nat)'.
Recommended: make explicit what you conclude,
by using 'We conclude that ...`.
No hint available.
COQC waterproof/test/reverse_engineer/auto_hintdb_arg.v
- : 'a list option = Some ([])
- : 'a list option = None
- : ident list option = Some ([@nocore])
COQC waterproof/test/string_auxiliary_test.v
Test passed: strings are equal
- : unit = ()
Test passed: strings are equal
- : unit = ()
Test passed: strings are equal
- : unit = ()
Test passed: strings are equal
- : unit = ()
- : bool = true
Test passed: received true
- : unit = ()
Test passed: received false
- : unit = ()
Test passed: received false
- : unit = ()
COQC waterproof/test/test_auxiliary_test.v
Test passed: lists indeed equal
- : unit = ()
Test passed: lists indeed equal
- : unit = ()
Unequal elements:12
Test passed, got error:TestFailedError ("List have different element")
- : unit = ()
Indeed hyp exists:n
Internal (err:(Hypothesis "x" not found))
Test passed, got error:TestFailedError ("Hyp not found")
Indeed hyp exists:n
Hypothesis 'n' indeed has type: nat
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (n = 1)
Internal (err:(Hypothesis "m" not found))
Test passed, got error:TestFailedError ("Hyp not found")
Indeed hyp exists:n
Expected type: bool, actual type: nat
Test passed, got error:TestFailedError ("Hyp has wrong type")
Test passed: received constr:(true)
- : unit = ()
```

```
Test passed, got error:TestFailedError
                         ("Did not get a constr equal to a bool with value true")
- : unit = ()
Test passed, got error:TestFailedError
                         ("Did not get a constr equal to a bool with value true")
- : unit = ()
Test passed: received true
- : unit = ()
Test passed, got error:TestFailedError
                         ("Expected Ltac2 true, got Ltac2 bool 'false'")
- : unit = ()
Test passed: received false
- : unit = ()
Test passed, got error:TestFailedError
                         ("Expected Ltac2 FALSE, got Ltac2 bool 'true'")
- : unit = ()
Target is indeed equal to the goal.
Test passed, got error:TestFailedError ("Target not equal to the goal.")
Constr indeed equal.
- : unit = ()
Constr not equal, got: (1 < 2) and: (1 > 2)
Test passed, got error:TestFailedError ("Constr not equal.")
- : unit = ()
COQC waterproof/test/test_contradiction_tactics/test_basic_contradiction.v
COQC waterproof/test/test_load_database.v
Initial database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBReals; WaterproofDBAdditional;
WaterproofDBSets; WaterproofDBRealsAndIntegers; WaterproofDBSquareRoot;
WaterproofDBPlusMinus; WaterproofDBExponential; WaterproofDBAbsoluteValue;
WaterproofDBZeroOne; WaterproofDBOther; WaterproofDBMultiplication]
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = []
Waterproof could not find a proof of (forall x y : R,
                                      (x + y) ^ 2 = x ^ 2 + y ^ 2 + 2 * x * y)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBMultiplication]
Waterproof could not find a proof of (forall x y : R, x + y = y + x)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBPlusMinus]
Waterproof could not find a proof of (forall x y : R,
                                      (x + y) * x = x * x + x * y)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBPlusMinus]
Waterproof could not find a proof of (forall x y : R, x * y = y * x)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBMultiplication]
Waterproof could not find a proof of (forall x y : R, x + x + y = x + y + x)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBZeroOne]
Waterproof could not find a proof of (forall x y : R, x * y = y * x)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBMultiplication]
Waterproof could not find a proof of (forall x : R,
                                      R -> x * 1 + 1 + 0 = x + 1)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBMultiplication]
Waterproof could not find a proof of (forall x y : R,
                                      (x + y) ^ 2 = x ^ 2 + y ^ 2 + 2 * x * y)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBSquareRoot]
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBReals; WaterproofDBAdditional;
WaterproofDBSets; WaterproofDBRealsAndIntegers; WaterproofDBSquareRoot;
WaterproofDBPlusMinus; WaterproofDBExponential; WaterproofDBAbsoluteValue;
WaterproofDBZeroOne; WaterproofDBOther; WaterproofDBMultiplication]
Waterproof could not find a proof of (forall x y : R, x * y = y * x + 0)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBZeroOne]
Waterproof could not find a proof of (forall x y : R, x * y = y * x + 0)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBZeroOne]
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBMultiplication;
WaterproofDBZeroOne]
Waterproof could not find a proof of (forall x y : R, x * y = y * x + 0)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBOther; WaterproofDBAbsoluteValue;
WaterproofDBSquareRoot; WaterproofDBPlusMinus; WaterproofDBMultiplication;
WaterproofDBZeroOne]
Waterproof could not find a proof of (forall a : R, ln (exp a) = a)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBExponential]
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBOther; WaterproofDBExponential;
WaterproofDBSquareRoot; WaterproofDBPlusMinus; WaterproofDBMultiplication;
WaterproofDBZeroOne; WaterproofDBExponential]
Waterproof could not find a proof of (forall a : R, Rabs a = Rabs (- a))
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current database selection is:
- : unit = ()
- : WaterproofDatabase list = [WaterproofDBAbsoluteValue]
Waterproof could not find a proof of (forall x y : R,
                                      (x + y) ^ 2 =
                                      Rabs (x ^ 2) + 2 * x * y + y ^ 2)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
COQC waterproof/test/test_set_intuition.v
Waterproof could not find a proof of (forall A B : Prop, A /\ B -> B /\ A)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
COQC waterproof/test/test_set_search_depth.v
Initial search depth is:
- : unit = ()
- : int = 2
Current search depth is:
- : unit = ()
- : int = 1
Waterproof could not find a proof of (forall x : R, x = 1 -> x = 2 -> x <> x)
Test passed, got error:AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Current search depth is:
- : unit = ()
- : int = 2
COQC waterproof/test/test_tactics/assume_test.v
File "/home/cosmin/Final SEP Branch/coq-waterproof/waterproof/tactics/assume.v", line 226,
    characters 4-26:
Warning: The following expression should have type unit. [not-unit,ltac]
File "/home/cosmin/Final SEP Branch/coq-waterproof/waterproof/tactics/assume.v", line 238,
    characters 4-31:
Warning: The following expression should have type unit. [not-unit,ltac]
File "./waterproof/test/test_tactics/assume_test.v", line 89, characters 4-91:
Warning: The following expression should have type unit. [not-unit,ltac]
Indeed hyp exists:c
Test passed: lists indeed equal
Test passed: lists indeed equal
File "./waterproof/test/test_tactics/assume_test.v", line 125, characters 4-71:
Warning: The following expression should have type unit. [not-unit,ltac]
Test passed: lists indeed equal
Test passed: received true
Test passed: received false
Test passed: received true
(2 <= x)
(2 <= x)
Test passed: received true
Testcases for [assume_premise_with_breakdown]
- : unit = ()
false
Hypotheses successfully assumed
false
Hypotheses successfully assumed
Indeed hyp exists:ab
Hypothesis 'ab' indeed has type: (A /\ B)
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
false
Hypotheses successfully assumed
Indeed hyp exists:a
Hypothesis 'a' indeed has type: A
Indeed hyp exists:b
Hypothesis 'b' indeed has type: B
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
Hypotheses successfully assumed
Indeed hyp exists:ab
Hypothesis 'ab' indeed has type: (A /\ B)
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
Testcases for Assume
- : unit = ()
false
Hypotheses successfully assumed
Indeed hyp exists:n_is_one
Hypothesis 'n_is_one' indeed has type: (n = 1)
Indeed hyp exists:n_is_two
Hypothesis 'n_is_two' indeed has type: (n = 2)
false
Hypotheses successfully assumed
Indeed hyp exists:ab
Hypothesis 'ab' indeed has type: (A /\ B)
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
false
Hypotheses successfully assumed
Indeed hyp exists:a
Hypothesis 'a' indeed has type: A
Indeed hyp exists:b
Hypothesis 'b' indeed has type: B
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
Hypotheses successfully assumed
Indeed hyp exists:ab
```

```
Hypothesis 'ab' indeed has type: (A /\ B)
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
false
Hypotheses successfully assumed
Indeed hyp exists:h_a
Hypothesis 'h_a' indeed has type: A
Indeed hyp exists:h_b
Hypothesis 'h_b' indeed has type: B
Indeed hyp exists:h_cd
Hypothesis 'h_cd' indeed has type: (C /\ D)
true
Hypotheses successfully assumed
Indeed hyp exists:n_is_one
Hypothesis 'n_is_one' indeed has type: (n = 1)
true
Hypotheses successfully assumed
Indeed hyp exists:x_ge_one
Hypothesis 'x_ge_one' indeed has type: (x > 1)
true
Hypotheses successfully assumed
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (some_function x = 3)
COQC waterproof/test/test_tactics/goal_to_hint_test.v
The goal has a forall-quantifier ().
You may need to introduce an arbitrary variable of type nat.
This can for example be done using 'Take ... : ...'.
Or you may need to make an assumption stating nat.
This can for example be done using 'Assume ... : ...'.
The goal has a forall-quantifier ().
You may need to introduce an arbitrary variable of type nat.
This can for example be done using 'Take ... : ...'.
Or you may need to make an assumption stating nat.
This can for example be done using 'Assume ... : ...'.
- : unit = ()
- : unit = ()
The goal has an implication ().
You may need to assume the premise
(0 = 0).
This can for example be done using 'Assume ... : ...'.
The goal has an implication ().
You may need to assume the premise
(0 = 0).
This can for example be done using 'Assume ... : ...'.
- : unit = ()
- : unit = ()
The goal has an existential quantifier ().
You may want to choose a specific variable of type nat.
This can for example be done using 'Choose ... '.
The goal has an existential quantifier ().
You may want to choose a specific variable of type nat.
This can for example be done using 'Choose ... '.
COQC waterproof/test/test_tactics/rewrite_inequalities_test.v
- : unit = ()
- : unit = ()
- : unit = ()
- : string = "ok"
- : string = "ok"
- : string = "ok"
- : string = "ok"
Test passed, got error:TestFailedError ("Wrong output given")
- : string = "ok"
File "./waterproof/test/test_tactics/rewrite_inequalities_test.v", line 116, characters
    4-109:
Warning: The following expression should have type unit. [not-unit,ltac]
- : unit = ()
- : unit = ()
File "./waterproof/test/test_tactics/rewrite_inequalities_test.v", line 140, characters 4-70:
Warning: The following expression should have type unit. [not-unit,ltac]
- : unit = ()
- : unit = ()
Constr indeed equal.
- : unit = ()
Constr indeed equal.
- : unit = ()
Constr indeed equal.
Constr indeed equal.
Constr indeed equal.
'<' does not work here.
with
h
and goal
(a > d)
Got error:
Tactic_failure (None)
```

```
Test passed, got error:RewriteError ("Failed to rewrite (in)equality")
Constr indeed equal.
COQC waterproof/test/test_tactics/such_that_test.v
File "/home/cosmin/Final SEP Branch/coq-waterproof/waterproof/tactics/assume.v", line 226,
    characters 4-26:
Warning: The following expression should have type unit. [not-unit,ltac]
File "/home/cosmin/Final SEP Branch/coq-waterproof/waterproof/tactics/assume.v", line 238,
    characters 4-31:
Warning: The following expression should have type unit. [not-unit,ltac]
false
Hypotheses successfully assumed
Indeed hyp exists:ab
Hypothesis 'ab' indeed has type: (A /\ B)
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
false
Hypotheses successfully assumed
Indeed hyp exists:a
Hypothesis 'a' indeed has type: A
Indeed hyp exists:b
Hypothesis 'b' indeed has type: B
Indeed hyp exists:bc
Hypothesis 'bc' indeed has type: (B /\ C)
true
Hypotheses successfully assumed
true
Hypotheses successfully assumed
Indeed hyp exists:x
Hypothesis 'x' indeed has type: nat
Indeed hyp exists:x_bigger_1
Hypothesis 'x_bigger_1' indeed has type: (x > 1)
COQC waterproof/test/test_tactics/take_test.v
Test passed, got error:TakeError
                         ("The type of the variable must match the type of the 'forall' goal'
                             s bound variable.")
Test passed, got error:TakeError
                         ("'Take' can only be applied to 'forall' goals")
Test passed, got error:TakeError
                         ("The type of the variable must match the type of the 'forall' goal'
                             s bound variable.")
Test passed, got error:Internal (err:(a is already used.))
Test passed, got error:TakeError
                         ("The type of the variable must match the type of the 'forall' goal'
                             s bound variable.")
Indeed hyp exists:A
Hypothesis 'A' indeed has type: subsets_R
Indeed hyp exists:a
Hypothesis 'a' indeed has type: (elements_R_satisfying (is_in A))
COQC waterproof/test/test_tactics/test_because.v
COQC waterproof/test/test_tactics/test_choose.v
Test passed, got error:ChooseError
                         ("'Choose' can only be applied to 'exists' goals")
Test passed, got error:ChooseError
                         ("'Choose' can only be applied to 'exists' goals")
COQC waterproof/test/test_tactics/test_choose_such_that.v
COQC waterproof/test/test_tactics/test_either.v
Recommendation: Please use comments to indicate the case
    you are in after writing this line. This helps to keep the proof readable.
Recommendation: Please use comments to indicate the case
    you are in after writing this line. This helps to keep the proof readable.
COQC waterproof/test/test_tactics/test_forward_reasoning/it_holds_that_test.v
- : WaterproofDatabase list = [WaterproofDBReals; WaterproofDBAdditional;
WaterproofDBSets; WaterproofDBRealsAndIntegers; WaterproofDBSquareRoot;
WaterproofDBPlusMinus; WaterproofDBExponential; WaterproofDBAbsoluteValue;
WaterproofDBZeroOne; WaterproofDBOther; WaterproofDBMultiplication]
Testcases for [By ... it holds that ... : ...]:
- : unit = ()
No hint available.
New sublemma successfully added.
Indeed hyp exists:this_lemma
Hypothesis 'this_lemma' indeed has type: (0 < 1)
No hint available.
Waterproof could not find a proof of (10 = 0)
Test passed, got error:AutomationFailure
                         ("Could not prove the given sublemma.
If you believe the statement should hold,
try making a smaller step.")
No hint available.
New sublemma successfully added.
No hint available.
Waterproof could not find a proof of (1 > 2)
Test passed, got error:AutomationFailure
                         ("Could not prove the given sublemma.
If you believe the statement should hold,
try making a smaller step.")
```

```
Testcases for [It holds that ... : ...]:
- : unit = ()
No hint available.
New sublemma successfully added.
Indeed hyp exists:this_lemma
Hypothesis 'this_lemma' indeed has type: True
No hint available.
Waterproof could not find a proof of (1 > 2)
Test passed, got error:AutomationFailure
                         ("Could not prove the given sublemma.
If you believe the statement should hold,
try making a smaller step.")
- : WaterproofDatabase list = [WaterproofDBReals; WaterproofDBAdditional;
WaterproofDBSets; WaterproofDBRealsAndIntegers; WaterproofDBSquareRoot;
WaterproofDBPlusMinus; WaterproofDBExponential; WaterproofDBAbsoluteValue;
WaterproofDBZeroOne; WaterproofDBOther; WaterproofDBMultiplication]
No hint available.
New sublemma successfully added.
COQC waterproof/test/test_tactics/test_forward_reasoning/it_suffices_to_show_test.v
The goal has an implication ().
You may need to assume the premise
(x > 0 /\ x < 2).
This can for example be done using 'Assume ... : ...'.
It indeed suffices to show that '(x = 1)'.
Target is indeed equal to the goal.
The goal has an implication ().
You may need to assume the premise
(A /\ A).
This can for example be done using 'Assume ... : ...'.
Waterproof could not find a proof of (A /\ A -> B)
Test passed, got error:AutomationFailure
                         ("Waterproof could not verify that this statement is enough to prove
                              the goal.")
It indeed suffices to show that '(0 = 0)'.
It indeed suffices to show that '(even 2)'.
Test passed, got error:AutomationFailure
                         ("Waterproof could not verify that this statement is enough to prove
                              the goal.")
COQC waterproof/test/test_tactics/test_forward_reasoning/proof_finishing_tactics_test.v
Test passed, got error:ReflexivityError
                         ("Reflexivity cannot be applied here.")
Recommended: make explicit what you conclude,
by using 'We conclude that ...`.
No hint available.
Recommended: make explicit what you conclude,
by using 'We conclude that ...`.
The goal has an implication ().
You may need to assume the premise A.
This can for example be done using 'Assume ... : ...'.
Recommended: make explicit what you conclude,
by using 'We conclude that ...`.
The goal has an implication ().
You may need to assume the premise A.
This can for example be done using 'Assume ... : ...'.
Waterproof could not find a proof of (A -> C)
Test passed, got error:waterprove.AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
Test passed, got error:AssumptionError
                         ("No hypothesis found that equals the current goal.")
Test passed, got error:AssumptionError
                         ("No hypothesis found that equals the current goal.")
The actual goal (A) is not equivalent to the goal you gave ((1 = 1)).
Test passed, got error:waterprove.AutomationFailure
                         ("Given goal not equivalent to actual goal.")
COQC waterproof/test/test_tactics/test_forward_reasoning/rewrite_using_test.v
Nat.mul_add_distr_l
     : forall n m p : nat, n * (m + p) = n * m + n * p
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
Target is indeed equal to the goal.
Nat.add_assoc
     : forall n m p : nat, n + (m + p) = n + m + p
Successfully rewrote goal using '(forall n m p : nat, n + (m + p) = n + m + p)'.
Target is indeed equal to the goal.
Test passed, got error:RewriteError
                         ("Could not rewrite goal with this expression")
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
Target is indeed equal to the goal.
Successfully rewrote goal using '(forall n m p : nat, n + (m + p) = n + m + p)'.
Target is indeed equal to the goal.
Test passed, got error:RewriteError
                         ("Could not rewrite goal with this expression")
```

```
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (5 * x + 5 * y = 10)
and_comm
     : forall A B : Prop, A /\ B <-> B /\ A
Waterproof could not find a proof of (forall A B : Prop, A /\ B <-> B /\ A)
Test passed, got error:AutomationFailure
                         ("Could not verify that the proposition used for the rewrite holds.
You may need to prove this proposition first before rewriting others with it.")
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (5 * (x + y) = 10)
Waterproof could not find a proof of (forall A B : Prop, A /\ B <-> B /\ A)
Test passed, got error:AutomationFailure
                         ("Could not verify that the proposition used for the rewrite holds.
You may need to prove this proposition first before rewriting others with it.")
COQC waterproof/test/test_tactics/test_forward_reasoning/test_apply.v
COQC waterproof/test/test_tactics/test_forward_reasoning/test_claims.v
COQC waterproof/test/test_tactics/test_forward_reasoning/test_define.v
COQC waterproof/test/test_tactics/test_forward_reasoning/test_induction.v
COQC waterproof/test/test_tactics/test_forward_reasoning/test_simplify.v
COQC waterproof/test/test_tactics/test_forward_reasoning/we_conclude_that_test.v
Testcases for [We conclude that ...]:
- : unit = ()
No hint available.
The actual goal (True) is not equivalent to the goal you gave (False).
Test passed, got error:waterprove.AutomationFailure
                         ("Given goal not equivalent to actual goal.")
Should raise warning:
- : unit = ()
Warning:
The statement you provided does not exactly correspond to what you need to show.
This can make your proof less readable.
Waterproof will try to rewrite the goal...
No hint available.
Should raise warning:
- : unit = ()
No hint available.
No hint available.
Waterproof could not find a proof of (0 = 1)
Test passed, got error:waterprove.AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
No hint available.
Testcases for [By ... we conclude that ...]:
- : unit = ()
No hint available.
The actual goal ((0 + 0 = 20)) is not equivalent to the goal you gave (
(0 + 0 = 10)).
Test passed, got error:waterprove.AutomationFailure
                         ("Given goal not equivalent to actual goal.")
No hint available.
Should raise warning:
- : unit = ()
Warning:
The statement you provided does not exactly correspond to what you need to show.
This can make your proof less readable.
Waterproof will try to rewrite the goal...
No hint available.
Should raise warning:
- : unit = ()
No hint available.
Waterproof could not find a proof of (0 = 1)
Test passed, got error:waterprove.AutomationFailure
                         ("Waterproof could not find a proof.
If you believe the statement should hold,
try making a smaller step.")
No hint available.
No hint available.
COQC waterproof/test/test_tactics/test_forward_reasoning/write_as_test.v
Successfully rewrote 'h' to (Aux.type_of h).
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (x = 3)
Test passed, got error:RewriteError
                         ("Cannot rewrite the hypothesis with this term.")
COQC waterproof/test/test_tactics/test_forward_reasoning/write_using_test.v
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
Indeed hyp exists:h
Hypothesis 'h' indeed has type: (5 * x + 5 * y = 10)
Successfully rewrote goal using '(forall n m p : nat,
                                  n * (m + p) = n * m + n * p)'.
```

```
Expected result of rewrite: '(5 * y + 5 * x = 10)'.
Actual result: '
(5 * x + 5 * y = 10)'.
Test passed, got error:RewriteError
                          ("Rewriting the hypothesis with this equality is possible,
but did produce the expected result")
Successfully rewrote goal using '(forall n m p : nat,
                                   n * m + n * p = n * (m + p))'.
Target is indeed equal to the goal.
Successfully rewrote goal using '(forall n m p : nat,
                                   n * m + n * p = n * (m + p))'.
Expected result of rewrite: '(5 * (x + y) = 11)'.
Actual result: '
(5 * (x + y) = 10)'.
Test passed, got error:RewriteError
                          ("Rewriting the hypothesis with this equality is possible,
but did produce the expected result")
COQC waterproof/test/test_tactics/test_sets_tactics/test_sets_automation_tactics.v
File "./waterproof/test/test_tactics/test_sets_tactics/test_sets_automation_tactics.v", line
    49, characters 0-29:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/test/test_tactics/test_sets_tactics/test_sets_automation_tactics.v", line
    49, characters 0-29:
Warning: The default value for hint locality is currently "local" in a
section and "global" otherwise, but is scheduled to change in a future
release. For the time being, adding hints outside of sections without
specifying an explicit locality is therefore deprecated. It is recommended to
use "export" whenever possible. [deprecated-hint-without-locality,deprecated]
File "./waterproof/test/test_tactics/test_sets_tactics/test_sets_automation_tactics.v", line
    51, characters 0-18:
Warning: Interpreting this declaration as if a global declaration prefixed by
"Local", i.e. as a global declaration which shall not be available without
qualification when imported. [local-declaration,scope]
File "./waterproof/test/test_tactics/test_sets_tactics/test_sets_automation_tactics.v", line
    52, characters 0-28:
Warning: Interpreting this declaration as if a global declaration prefixed by
"Local", i.e. as a global declaration which shall not be available without
qualification when imported. [local-declaration,scope]
No hint available.
Waterproof could not find a proof of (In U (Empty_set U) x)
No hint available.
No hint available.
No hint available.
No hint available.
No hint available.
No hint available.
No hint available.
No hint available.
No hint available.
COQC waterproof/test/test_tactics/test_we_know.v
We indeed know that (x = 3)
We indeed know that (n = m)
We indeed know that (forall n : nat, n = n)
Test passed, got error:WeKnowError ("This hypothesis does not exist.")
Test passed, got error:Internal (err:(Hypothesis "Z" not found))
We indeed know that (x + (x + 0) = 4)
COQC waterproof/test/test_tactics/test_we_show_both_directions.v
Test passed, got error:BothDirectionsError
                          ("This is not an if and only if, so try another tactic.")
COQC waterproof/test/test_tactics/test_we_show_both_statements.v
Test passed, got error:BothStatementsError
                          ("This is not an 'and' statement, so try another tactic.")
You need to show (n + 1 = n + 1) instead of (n + 2 = n + 2)
You need to show (n = n) instead of (n + 2 = n + 2)
You need to show (n + 1 = n + 1) instead of (n + 2 = n + 2)
Test passed, got error:BothStatementsError
                          ("None of these two statements are what you need to show.")
COQC waterproof/test/test_tactics/unfold_test.v
COQC waterproof/test/test_tactics/we_need_to_show_test.v
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (1 = 1)
The goal is indeed: (double 2 = 4)
The goal is indeed: (2 * 2 = 4)
Test passed, got error:GoalCheckError ("No such goal")
```

As we can see from the above results, there are no errors after running the `make` operation (only warnings, but they are not important).

6 Test coverage

6.1 Installer

In total there are four files:

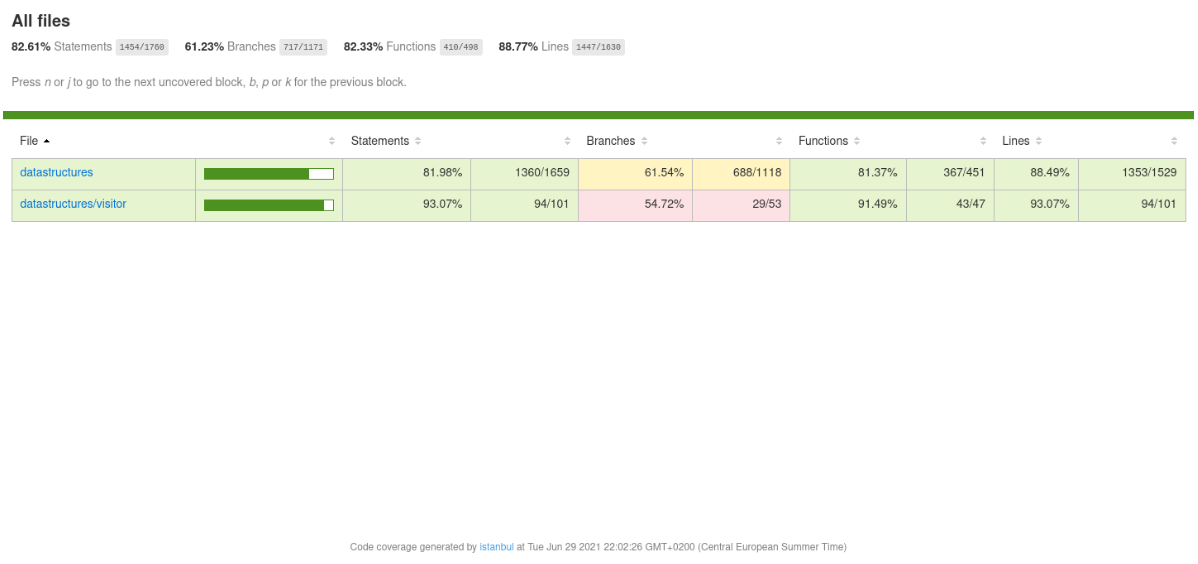
- `patch_serapi.sh`
- `install_packages.sh`
- `unselect_package.sh`
- `install_custom_nsis.sh`

The `patch_serapi.sh` file does not have unit test cases. This is also the smallest file only containing calls to the `sed` stream editor. This is a very simple file where tests would be more error-prone than the code itself. As such, we deemed this file as **not** requiring unit test cases.

When running the unit tests, all lines of the last three files were executed. Hence, code coverage of these files is 100%. Notice that since there is not a lot of logic in the shell files, this is easy to achieve.

6.2 Abstract Syntax Tree

The unit test coverage is shown in Figure 2 and Figure 3. The reports are generated using the Mocha test framework and the Istanbul test coverage tool.



All files

82.61% Statements 1454/1760 **61.23%** Branches 717/1171 **82.33%** Functions 410/498 **88.77%** Lines 1447/1630

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

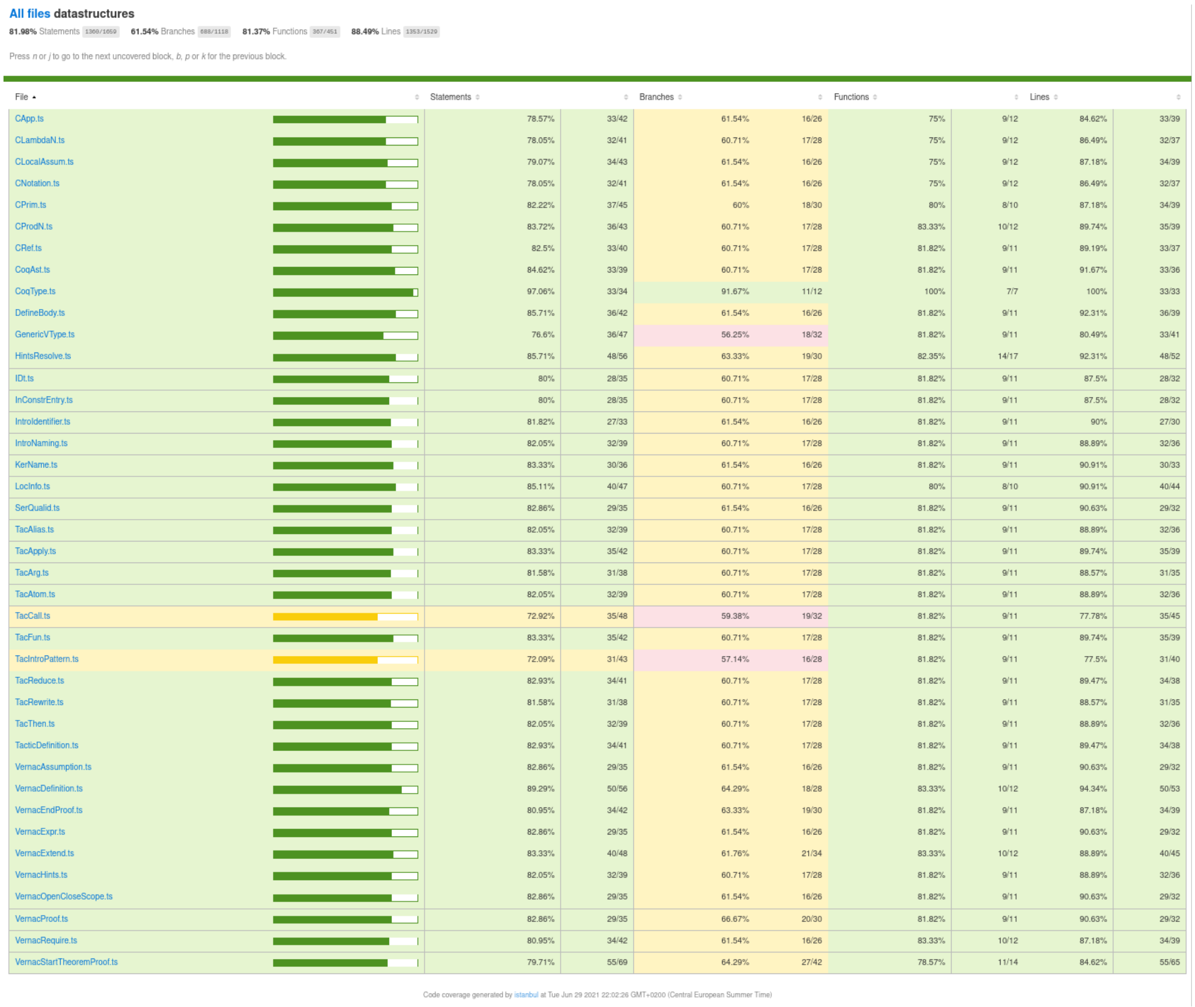
File		Statements		Branches		Functions		Lines	
datastructures		81.98%	1360/1659	61.54%	688/1118	81.37%	367/451	88.49%	1353/1529
datastructures/visitor		93.07%	94/101	54.72%	29/53	91.49%	43/47	93.07%	94/101

Code coverage generated by istanbul at Tue Jun 29 2021 22:02:26 GMT+0200 (Central European Summer Time)

Figure 2: Overview of the AST code coverage report

The results were obtained using the following terminal command:

```
npm run test-ast:cov
```



All files datastructures

81.98% Statements 1360/1659 **61.54%** Branches 688/1118 **81.37%** Functions 367/451 **88.49%** Lines 1353/1529

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
CApp.ts	78.57%	33/42	61.54%	16/26	75%	9/12	84.62%	33/39
CLambdaN.ts	78.05%	32/41	60.71%	17/28	75%	9/12	86.49%	32/37
CLocalAssum.ts	79.07%	34/43	61.54%	16/26	75%	9/12	87.18%	34/39
CNotation.ts	78.05%	32/41	61.54%	16/26	75%	9/12	86.49%	32/37
CPrim.ts	82.22%	37/45	60%	18/30	80%	8/10	87.18%	34/39
CProdN.ts	83.72%	36/43	60.71%	17/28	83.33%	10/12	89.74%	35/39
CRef.ts	82.5%	33/40	60.71%	17/28	81.82%	9/11	89.19%	33/37
CoqAst.ts	84.62%	33/39	60.71%	17/28	81.82%	9/11	91.67%	33/36
CoqType.ts	97.06%	33/34	91.67%	11/12	100%	7/7	100%	33/33
DefineBody.ts	85.71%	36/42	61.54%	16/26	81.82%	9/11	92.31%	36/39
GenericVType.ts	76.6%	36/47	56.25%	18/32	81.82%	9/11	80.49%	33/41
HintsResolve.ts	85.71%	48/56	63.33%	19/30	82.35%	14/17	92.31%	48/52
IDt.ts	80%	28/35	60.71%	17/28	81.82%	9/11	87.5%	28/32
InConstrEntry.ts	80%	28/35	60.71%	17/28	81.82%	9/11	87.5%	28/32
IntroIdentifier.ts	81.82%	27/33	61.54%	16/26	81.82%	9/11	90%	27/30
IntroNaming.ts	82.05%	32/39	60.71%	17/28	81.82%	9/11	88.89%	32/36
KerName.ts	83.33%	30/36	61.54%	16/26	81.82%	9/11	90.91%	30/33
LocInfo.ts	85.11%	40/47	60.71%	17/28	80%	8/10	90.91%	40/44
SerQualid.ts	82.86%	29/35	61.54%	16/26	81.82%	9/11	90.63%	29/32
TacAlias.ts	82.05%	32/39	60.71%	17/28	81.82%	9/11	88.89%	32/36
TacApply.ts	83.33%	35/42	60.71%	17/28	81.82%	9/11	89.74%	35/39
TacArg.ts	81.58%	31/38	60.71%	17/28	81.82%	9/11	88.57%	31/35
TacAtom.ts	82.05%	32/39	60.71%	17/28	81.82%	9/11	88.89%	32/36
TacCall.ts	72.92%	35/48	59.38%	19/32	81.82%	9/11	77.78%	35/45
TacFun.ts	83.33%	35/42	60.71%	17/28	81.82%	9/11	89.74%	35/39
TacIntroPattern.ts	72.09%	31/43	57.14%	16/28	81.82%	9/11	77.5%	31/40
TacReduce.ts	82.93%	34/41	60.71%	17/28	81.82%	9/11	89.47%	34/38
TacRewrite.ts	81.58%	31/38	60.71%	17/28	81.82%	9/11	88.57%	31/35
TacThen.ts	82.05%	32/39	60.71%	17/28	81.82%	9/11	88.89%	32/36
TacticDefinition.ts	82.93%	34/41	60.71%	17/28	81.82%	9/11	89.47%	34/38
VernacAssumption.ts	82.86%	29/35	61.54%	16/26	81.82%	9/11	90.63%	29/32
VernacDefinition.ts	89.29%	50/56	64.29%	18/28	83.33%	10/12	94.34%	50/53
VernacEndProof.ts	80.95%	34/42	63.33%	19/30	81.82%	9/11	87.18%	34/39
VernacExpr.ts	82.86%	29/35	61.54%	16/26	81.82%	9/11	90.63%	29/32
VernacExtend.ts	83.33%	40/48	61.76%	21/34	83.33%	10/12	88.89%	40/45
VernacHints.ts	82.05%	32/39	60.71%	17/28	81.82%	9/11	88.89%	32/36
VernacOpenCloseScope.ts	82.86%	29/35	61.54%	16/26	81.82%	9/11	90.63%	29/32
VernacProof.ts	82.86%	29/35	66.67%	20/30	81.82%	9/11	90.63%	29/32
VernacRequire.ts	80.95%	34/42	61.54%	16/26	83.33%	10/12	87.18%	34/39
VernacStartTheoremProof.ts	79.71%	55/69	64.29%	27/42	78.57%	11/14	84.62%	55/65

Code coverage generated by istanbul at Tue Jun 29 2021 22:02:26 GMT+0200 (Central European Summer Time)

Figure 3: In-depth AST code coverage report

After running the code coverage, an overall coverage of 81.98% is obtained for the CoqType data structures and 93% on the `FlattenVisitor` implementation.

As export statements, which allow for a class or function to be visible from other files and TypeScript type definitions cannot be covered by the test coverage, and certain behaviours such as syntax highlighting do not render themselves to unit tests, we are satisfied with this coverage result.

6.3 New Tactics Library

As previously mentioned, Coq is an interactive proof assistant, and $L_{tac}2$ is a functional programming language. None of them are designed for testing purposes, and so there is no automatic tooling that determines the amount of code the unit tests presented in Section 3.4 cover. Hence, determining the test coverage must be done *manually*. We will manually check the test coverage not by lines of code tested (due to obvious reasons), but by the number of files tested. This is because the testing (and the software overall) was implemented in the following way:

- The tactics library contains a collection of auxiliary testing functions. This is the file called `test_auxiliary.v` which is also unit tested on its own. These functions compare an observed and an expected value, and raise an error if and only if the result is not as expected.

- (Almost) Every source-code file has a test-code file counterpart. These test files consist of a series of tests that depend on the auxiliary testing functions. Since these functions raise an error if any test fails, the test files can only be compiled if all tests pass.

The relevant file structure (i.e. the files and their folder structure) is the following:

```
|  🟥 _CoqProject
|__🟥 Makefile
|__🟥 waterproof
     |__🟦 waterprove
     |    |__🟩 waterprove.v
     |__🟦 AllTactics.v
     |__🟦 auxiliary.v
     |__🟦 contradiction_tactics
     |    |__🟩 basic_contradiction.v
     |__🟦 databases.v
     |__🟦 definitions
     |    |__🟩 set_definitions.v
     |    |__🟩 subsequences_definitions.v
     |__🟦 deprecated
     |    |__🟩 databases_test.v
     |    |__🟩 deprecated_collect_db.v
     |    |__🟩 load_databases_demo.v
     |    |__🟩 test_unfold.v
     |__🟦 load_database
     |    |__🟩 AbsoluteValue.v
     |    |__🟩 Additional.v
     |    |__🟩 All.v
     |    |__🟩 DisableWildcard.v
     |    |__🟩 EnableWildcard.v
     |    |__🟩 Exponential.v
     |    |__🟩 Multiplication.v
     |    |__🟩 Other.v
     |    |__🟩 PlusMinus.v
     |    |__🟩 reals.v
     |    |__🟩 Sets.v
     |    |__🟩 SquareRoot.v
     |    |__🟩 ZeroOne.v
     |__🟦 notations
     |    |__🟩 notations.v
     |    |__🟩 set_notations.v
     |__🟦 selected_databases.v
     |__🟦 set_intuition
     |    |__🟩 Disabled.v
     |    |__🟩 Enabled.v
     |__🟦 set_search_depth
          |__🟩 To_1.v
          |__🟩 To_2.v
          |__🟩 To_3.v
```

```
│
│   ├── 🟩 To_4.v
│   └── 🟩 To_5.v
├── 🟦 string_auxiliary.v
├── 🟦 tactics
│   ├── 🟩 assume.v
│   ├── 🟩 automation_databases
│   │   └── 🟧 decidability_db.v
│   ├── 🟩 because.v
│   ├── 🟩 choose_such_that.v
│   ├── 🟩 choose.v
│   ├── 🟩 either.v
│   ├── 🟩 goal_to_hint.v
│   ├── 🟩 rewrite_inequalities.v
│   ├── 🟩 take.v
│   ├── 🟩 unfold.v
│   ├── 🟩 we_know.v
│   ├── 🟩 we_need_to_show.v
│   ├── 🟩 we_show_both_directions.v
│   ├── 🟩 we_show_both_statements.v
│   ├── 🟩 sets_tactics
│   │   └── 🟧 sets_automation_tactics.v
│   └── 🟩 forward_reasoning
│       ├── 🟧 apply.v
│       ├── 🟧 claims.v
│       ├── 🟧 define.v
│       ├── 🟧 forward_reasoning_aux.v
│       ├── 🟧 induction.v
│       ├── 🟧 it_holds_that.v
│       ├── 🟧 it_suffices_to_show.v
│       ├── 🟧 proof_finishing_tactics.v
│       ├── 🟧 rewrite_using.v
│       ├── 🟧 simplify.v
│       ├── 🟧 we_conclude_that.v
│       ├── 🟧 write_as.v
│       └── 🟧 write_using.v
└── 🟦 test
    ├── 🟩 reverse_engineer
    │   └── 🟧 auto_hintdb_arg.v
    ├── 🟩 test_contradiction_tactics
    │   └── 🟧 test_basic_contradiction.v
    ├── 🟩 string_auxiliary_test.v
    ├── 🟩 test_auxiliary_test.v
    ├── 🟩 test_load_database.v
    ├── 🟩 test_set_intuition.v
    ├── 🟩 test_set_search_depth.v
    ├── 🟩 test_auxiliary.v
    └── 🟩 test_tactics
        ├── 🟧 assume_test.v
        ├── 🟧 goal_to_hint_test.v
        └── 🟧 rewrite_inequalities_test.v
```

```
│
├── 🟧 such_that_test.v
├── 🟧 take_test.v
├── 🟧 test_because.v
├── 🟧 test_choose_such_that.v
├── 🟧 test_choose.v
├── 🟧 test_either.v
├── 🟧 test_we_know.v
├── 🟧 test_we_show_both_directions.v
├── 🟧 test_we_show_both_statements.v
├── 🟧 unfold_test.v
├── 🟧 we_need_to_show_test.v
├── 🟧 test_sets_tactics
│   └── 🟨 test_sets_automation_tactics.v
└── 🟧 test_forward_reasoning
    ├── 🟨 it_holds_that_test.v
    ├── 🟨 it_suffices_to_show_test.v
    ├── 🟨 proof_finishing_tactics_test.v
    ├── 🟨 rewrite_using_test.v
    ├── 🟨 test_apply.v
    ├── 🟨 test_claims.v
    ├── 🟨 test_define.v
    ├── 🟨 test_induction.v
    ├── 🟨 test_simplify.v
    ├── 🟨 we_conclude_that_test.v
    ├── 🟨 write_as_test.v
    └── 🟨 write_using_test.v
```

The files `_CoqProject` and `Makefile` are files used to run the `make` operation mentioned in Section 4.3.

The `AllTactics.v` file just exports all files under the `tactics` folder. So this is implicitly tested when testing the files under the `tactics` folder.

The contents of the `definitions` and `notations` folders consist of files containing various lemmas and their proofs, and are compiled by the `make` operation. While these files are used in the tactics library, they do not contain any functions. Moreover, determining whether or not the files compile is done after running the `make` operation. Hence, these files do not need separate unit tests, such as the ones described in Section 3.4.

The contents of the folder `deprecated` consist of files that are not compiled with the `make` operation, as they were failed experiments. They are therefore **not** used in the tactics library, and so they need not be unit tested.

Lastly, the contents of the folder `reverse_engineer` are comprised again of an experimental file that is not compiled using the `make` operation, and so this also need not be unit tested.

This leaves a total of 58 files that need to be unit tested (all files except the ones mentioned above and except the ones under the `test` folder). Out of these, considering the file structure and the unit tests mentioned in Section 3.4, only three files are not unit tested: `auxiliary.v`, `EnableWildcard.v` and `DisableWildcard.v`. This gives a total of 55 files that are unit tested, which amounts to $\approx 94.82\%$ code coverage by unit tests.