SpecCheck – Specification-Based Testing for Isabelle/ML

Kevin Kappelmann, Lukas Bulwahn, and Sebastian Willenbrink

September 13, 2023

Abstract

SpecCheck is a QuickCheck-like testing framework for Isabelle/ML. You can use it to write specifications for ML functions. SpecCheck then checks whether your specification holds by testing your function against a given number of generated inputs. It helps you to identify bugs by printing counterexamples on failure and provides you timing information.

SpecCheck is customisable and allows you to specify your own input generators, test output formats, as well as pretty printers and shrinking functions for counterexamples among other things.

Contents

1 SpecCheck Base

theory *SpecCheck-Base*
imports *Pure*
begin

Summary Basic setup for SpecCheck.

ML-file ‹*util.ML*›

ML-file ‹*speccheck-base.ML*›
ML-file ‹*property.ML*›
ML-file ‹*configuration.ML*›

ML-file ‹*random.ML*›

end

2 Generators

theory *SpecCheck-Generators*
imports *SpecCheck-Base*
begin

Summary Generators for SpecCheck take a state and return a pair consisting of a generated value and a new state. Refer to `gen_base.ML` for the most basic combinators.

ML-file ‹*gen-types.ML*›
ML-file ‹*gen-base.ML*›
ML-file ‹*gen-text.ML*›
ML-file ‹*gen-int.ML*›
ML-file ‹*gen-real.ML*›
ML-file ‹*gen-function.ML*›
ML-file ‹*gen-term.ML*›
ML-file ‹*gen.ML*›

end

3 Show

theory *SpecCheck-Show*
imports *Pure*
begin

Summary Show functions (pretty-printers) for SpecCheck take a value and return a `Pretty.T` representation of the value. Refer to `show_base.ML` for some basic examples.

ML-file ‹*show-types.ML*›
ML-file ‹*show-base.ML*›
ML-file ‹*show-term.ML*›
ML-file ‹*show.ML*›

end

4 Output Styles

theory *SpecCheck-Output-Style*
imports
SpecCheck-Base
SpecCheck-Show
begin

Summary Output styles for SpecCheck take the result of a test run and process it accordingly, e.g. by printing it or storing it to a file.

ML-file ‹*output-style-types.ML*›
ML-file ‹*output-style-perl.ML*›
ML-file ‹*output-style-custom.ML*›
ML-file ‹*output-style.ML*›

end

5 Shrinkers

theory *SpecCheck-Shrink*
imports *SpecCheck-Generators*
begin

Summary Shrinkers for SpecCheck take a value and return a sequence of smaller values derived from it. Refer to `shrink_base.ML` for some basic examples.

ML-file ‹*shrink-types.ML*›
ML-file ‹*shrink-base.ML*›
ML-file ‹*shrink.ML*›

end

6 SpecCheck

theory *SpecCheck*
imports
SpecCheck-Generators
SpecCheck-Show
SpecCheck-Shrink
SpecCheck-Output-Style
begin

Summary The SpecCheck (specification based) testing environment and Lecker testing framework.

ML-file ‹*lecker.ML*›
ML-file ‹*speccheck.ML*›

end

7 Dynamic Generators

theory *SpecCheck-Dynamic*
imports *SpecCheck*
begin

Summary Generators and show functions for SpecCheck that are dynamically derived from a given ML input string. This approach can be handy to quickly test a function during development, but it lacks customisability and is very brittle. See `../Examples/SpecCheck_Examples.thy` for some examples contrasting this approach to the standard one (specifying generators as ML code).

ML-file ‹*dynamic-construct.ML*›
ML-file ‹*speccheck-dynamic.ML*›

end