hypothesis-ros Documentation

Release 0.1.0

Florian Kromer

Aug 20, 2018

Contents:

hypothesis-ros looks small and cute. But it is one of those libraries on the dark side of power. It can be very powerful and very nasty. It enables property-based testing on the ROS node level. It provides generators for message data fields, parameters and message fields of a growing number of messages.

hypothesis-ros depends on property based testing framework hypothesis. The documentation for this library can be found in the hypothesis documentation.

The naming of strategies is according to ROS notation to ease the mapping of the strategies to the corresponding ROS data types, messages and parameters. Consider that this implies some conflict with Python builtins (e.g. *bool*, *list*).

CHAPTER 1

User's Guide

1.1 Compatibility with Python interpreters

hypothesis-ros uses *hypothesis* which implies the compatibility with Python interpreters. Hypothesis is supported and tested on CPython 2.7 and CPython 3.4+ (hypothesis docs python versions).

1.2 Compatibility with Python test frameworks (test runners)

hypothesis-ros uses *hypothesis* which implies the compatibility with Python test frameworks. Hypothesis is compatible with

- *unittest* (supported, tested, no limitations),
- *pytest* (supported, tested, limitations: function based fixtures do not behave like expected),
- *nose* (supported, tested, yield based tests do not work)

(hypothesis docs testing frameworks).

1.3 Configuration of settings

hypothesis was initially designed for Python source code level testing. Therefore the configuration of *settings* needs special care.

deadline: A deadline has either set to a very high value or should be disabled.

perform_health_checks: In case *perform_health_checks* is enabled some health checks need to be selectively disabled with *suppress_health_check*.

suppress_health_check: Refer to *Configuration of health checks*.

use_coverage: *hypothesis* supports coverage based data generation when the tests are executed on the Python source code level. *hypothesis-ros* does not support coverage based data generation. Enabling it has no effect/could raise errors.

A typical configuration of *timeout* and *deadline* in *@settings* looks like follows:

```
...
from hypothesis import settings, unlimited

...
@settings(timeout=unlimited,
          deadline=None,
          ...)
def test_node_does_not_crash(...):
    ...
```

The settings *database_file*, *database*, *buffer_size*, *derandomize*, *max_examples*, *max_iterations*, *max_shrinks*, *min_satisfying_examples*, *phases*, *stateful_step_count*, *strict*, ‘‘ usually don’t need special consideration and may be used as usual.

1.4 Configuration of health checks

hypothesis was initially designed for Python source code level testing. Therefore the configuration of *health checks* (hypothesis docs health checks) needs special care. In case health checks are performed (*perform_health_checks*) the health ckeck *too_slow* and *hung_test* need to be disabled via *suppress_healthcheck* usually.

A typical configuration of *suppress_health_check* in *@settings* looks like follows:

```
...
from hypothesis import settings, HealthCheck

...
@settings(...,
          suppress_health_check=[HealthCheck.too_slow,
                                 HealthCheck.hung_test]
         )
def test_node_does_not_crash(...):
    ...
```

The health checks *data_too_large*, *filter_too_much*, *return_value* and *large_base_example* don’t need special consideration and may be used as usual.

1.5 Configuration of example database

If a test fails hypothesis saves the test input in a atabase. The next time hypothesis runs this conditions will be used first. The configuration of the example database may be adjusted as usual (hypothesis docs example database).

CHAPTER 2

Changelog

2.1 v0.2.1

- fix invalid classifier for License

2.2 v0.2.0

- add strategies for - CompressedImage.msg - DisparityImage.msg - Image.msg - Imu.msg - PoseWithCovariance.msg - RegionOfInterest.msg - TFMessage.msg - TransformStamped.msg
- fix field frame_id in strategy Header.msg
- add validation in message field strategies
- refactor dublication of parameter strategies
- add missing tests for geometry_msgs
- add and cleanup tox environments
- change license to Apache 2
- fix docstrings
- improve docs

2.3 v0.1.0

Initial quick and dirty implementation of hypothesis strategies for ROS msg field types, ROS parameter types and ROS msgs.

buildin msg field types:

- bool

- int8
- uint8
- int16
- uint16
- int32
- uint32
- int64
- uint64
- float32
- float64
- string
- time
- duration
- array

parameter types:

- bool
- int32
- string
- date
- list
- double

std_msgs:

- Header

geometry_msgs:

- Point
- Quaternion
- Pose
- Transform
- Vector3

CHAPTER 3

Indices and tables

- genindex
- modindex
- search