hypothesis-ros Documentation

Release 1.0.0

Florian Kromer

Sep 30, 2019

Contents

Data generators for Property Based Testing and Fuzzy Testing of ROS1 nodes.

Warning: This repository is unmaintained and has been archived. Refer to ros1_fuzzer instead.

hypothesis-ros provides data generators for message data fields, parameters and common message fields of the Robot Operating System 1 (ROS). *hypothesis-ros* enables Property Based Testing and Fuzzy Testing of ROS1 nodes.

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

The naming of strategies is according to ROS1 notation to ease the mapping of the strategies to the corresponding ROS1 data types, messages and parameters. Consider that this implies some conflict with Python builtins (e.g. *bool*, *list*).

CHAPTER 1

User's Guide

This section is about configuration, integration and usage considerations.

1.1 Usage

The examples give a first (very limited) overview how to use *hypothesis-ros*. For further information about how to use the data generators of this package refer to the API documentation and the tests directory of the package source code repository.

1.1.1 Minimal Example

```
 pip install ipython
 ipython
In [1]: from hypothesis_ros.message_fields import int16
In [2]: int16().example()
Out[2]:-32183
In [3]: int16(min_value=5, max_value=5).example()
Out[3]: 5
```

1.1.2 Jupyter Notebook Examples

```
pip install jupyter
cd docs/source/notebooks/
jupyter lab
```

1.2 Compatibility

The ROS1 test infrastructure is limited w.r.t. integration of test tools into the build tooling. The recommended way to integrate *hypothesis-ros* with ROS1 is by implementing a container level test framework. pytest and nose2 (no "prove in use") are suitable test runner choices. docker, the official ROS1 docker images and docker-py may be used to provide the ROS1 node "runtime environment".

1.2.1 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). However *hypothesis-ros* is only tested against the main ROS1 Python version v2.7.

1.2.2 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

1.3.1 hypothesis 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 *hypothesis 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.

1.3.2 hypothesis 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.

1.3.3 hypothesis 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

hypothesis_ros

2.1 hypothesis_ros package

2.1.1 Subpackages

hypothesis_ros.converters package

Submodules

hypothesis_ros.converters.geometry_msgs module

hypothesis_ros.converters.sensor_msgs module

hypothesis_ros.converters.std_msgs module

Module contents

hypothesis_ros.messages package

Submodules

hypothesis_ros.messages.geometry_msgs module

hypothesis_ros.messages.sensor_msgs module

hypothesis_ros.messages.std_msgs module

hypothesis_ros.messages.stereo_msgs module

hypothesis_ros.messages.tf2_msgs module

Module contents

2.1.2 Submodules

2.1.3 hypothesis_ros.message_fields module

2.1.4 hypothesis_ros.parameters module

2.1.5 Module contents

CHAPTER 3

About hypothesis-ros

In April 2018 began the implementation of the low level library *hypothesis-ros* to lay a foundation to enable robustness testing of ROS1 C++ nodes/nodelets using a property based (instead of an example based) approach. This package provides ROS1 independent functionality to generate pseudo-random data for ROS1 message fields and ROS1 parameters. In the beginning *hypothesis-ros* contained Python dependent functionality only. The ROS1 dependent functionality was separated into the package *rospbt* (not existing anymore). Because the test code depends on ROS1 messages in some point in time anyway *rospbt* was merged into *hypothesis-ros*. All message package specific generators (geometry msgs, sensor msgs , etc.) have been migrated into *hypothesis-ros* as well. The easier package handling of separate packages (with *hypothesis-ros* having Python-only dependencies only) in the beginning has not outweighted increased maintenance effort due to separate packages and all implications going along with it.

CHAPTER 4

Changelog

4.1 v1.0.0

- add rospy converters for std_msgs Header
- add rospy converters for geometry_msgs
- Point
- Quaternion
- Pose
- Transform
- TransformStamped
- Vector3
- add rospy converters for sensor_msgs
- Imu
- Image
- fix typo in geometry_msgs _TransformStamped
- add test coverage analysis
- add CI tool integration
- add source code docs

4.2 v0.3.0

- add strategies for
- CameraInfo.msg

- PoseWithCovarianceStamped.msg
- improve tests

4.3 v0.2.1

- fix invalid classifier for License

4.4 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

4.5 v0.1.0

Initial quick and dirty implementation of hypothesis strategies for ROS1 msg field types, ROS1 parameter types and ROS1 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 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

H