Spade-BDI Documentation

Release 0.3.0

Sergio Frayle Pérez

Jun 13, 2023

Contents:

CHAPTER 1

Spade-BDI

Create hybrid agents with a BDI layer for the SPADE MAS Platform.

- Free software: GNU General Public License v3
- Documentation: https://spade-bdi.readthedocs.io. (to be completed)

1.1 Features

- Create agents that parse and execute an ASL file written in AgentSpeak.
- Supports Agentspeak-like BDI behaviours.
- Add custom actions and functions.
- Send TELL, UNTELL and ACHIEVE KQML performatives.

1.2 Examples

basic.py:

```
import getpass
from spade_bdi.bdi import BDIAgent

server = input("Please enter the XMPP server address: ")
password = getpass.getpass("Please enter the password: ")

a = BDIAgent("BasicAgent@" + server, password, "basic.asl")
a.start()

a.bdi.set_belief("car", "blue", "big")
a.bdi.print_beliefs()

print(a.bdi.get_belief("car"))
a.bdi.print_beliefs()

a.bdi.remove_belief("car", 'blue', "big")
a.bdi.print_beliefs()

print(a.bdi.get_beliefs())
a.bdi.set_belief("car", 'yellow')
```

basic.asl:

```
!start.

+!start <-
    +car(red);
    .a_function(3,W);
    .print("w =", W);
    literal_function(red,Y);
    .print("Y =", Y);
    .custom_action(8);
    +truck(blue).

+car(Color)
 <- .print("The car is ",Color).
```

1.3 Examples

basic.py:

```
import getpass
from spade_bdi.bdi import BDIAgent

server = input("Please enter the XMPP server address: ")
password = getpass.getpass("Please enter the password: ")

a = BDIAgent("BasicAgent@" + server, password, "basic.asl")
a.start()

a.bdi.set_belief("car", "blue", "big")
a.bdi.print_beliefs()

print(a.bdi.get_belief("car"))
a.bdi.print_beliefs()
```

(continues on next page)

(continued from previous page)

```
a.bdi.remove_belief("car", 'blue', "big")
a.bdi.print_beliefs()

print(a.bdi.get_beliefs())
a.bdi.set_belief("car", 'yellow')
```

basic.asl:

```
!start.

+!start <-
    +car(red);
    .a_function(3,W);
    .print("w =", W);
    literal_function(red,Y);
    .print("Y =", Y);
    .custom_action(8);
    +truck(blue).

+car(Color)
 <- .print("The car is ",Color).
```

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Spade-BDI, run this command in your terminal:

```
$ pip install spade_bdi
```

This is the preferred method to install Spade-BDI, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Spade-BDI can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/javipalanca/spade_bdi
```

Or download the tarball:

```
$ curl  -OL https://github.com/javipalanca/spade_bdi/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

3.1 Create custom internal actions and functions

You must to overload the `add_custom_actions` method and to use the `add_function` or `add` (for actions) decorator. This custom method receives always the `actions` parameter:

```
import spade_bdi

class MyCustomBDIAgent(BDIAgent):

    def add_custom_actions(self, actions):
        @actions.add_function(".my_function", (int,))
        def _my_function(x):
            return x * x

        @actions.add(".my_action", 1)
        def _my_action(agent, term, intention):
            arg = agentspeak.grounded(term.args[0], intention.scope)
            print(arg)
            yield
```

Hint: Adding a function requires to call the `add_function` decorator with two parameters: the name of the function (starting with a dot) and a tuple with the types of the parameters (e.g. `(int, str)`).

Hint: Adding an action requires to call the `add` decorator with two parameters: the name of the action (starting with a dot) and the number of parameters. Also, the method being decorated receives three parameters: `agent`, `term`, and `intention`.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/javipalanca/spade_bdi/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Spade-BDI could always use more documentation, whether as part of the official Spade-BDI docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/javipalanca/spade_bdi/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *spade_bdi* for local development.

1. Fork the *spade_bdi* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/spade_bdi.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv spade_bdi
$ cd spade_bdi/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 spade_bdi tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/javipalanca/spade_bdi/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_spade_bdi
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Sergio Frayle Pérez <sfp932705@gmail.com>

5.2 Contributors

- Javi Palanca <jpalanca@gmail.com>

CHAPTER 6

History

6.1 0.3.0 (2023-06-13)

- Updated to SPADE 3.3.0.

6.2 0.2.2 (2022-06-03)

- Added exception when belief is not initialized.
- Improved examples.
- Improved documentation.

6.3 0.2.1 (2020-04-13)

- Fixed a bug when updating beliefs.
- Upgraded spade version to 3.1.4.

6.4 0.2.0 (2020-02-24)

- Created add_custom_actions method.
- Added example for actions.
- Improved documentation.
- Added some helpers like pause_bdi, resume_bdi.
- Now the asl file in the constructor is mandatory.

6.5 0.1.4 (2019-07-10)

- Allow to send messages to JIDs stored as beliefs.

6.6 0.1.3 (2019-07-08)

- Allow .send to a list of receivers.
- Allow to receive messages with lists of lists.
- Fixed readme.

6.7 0.1.1 (2019-06-18)

- Moved from pyson to python-agentspeak
- Added some helpers like pause_bdi, resume_bdi.
- Now the asl file in the constructor is mandatory.
- Allow to send tell messages with no args.
- Allow sending messages with variables.
- Extended the examples.

6.8 0.1.0 (2019-03-09)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search