Evaluation Experiments

This folder contains the ABS models and scripts which have been used to conduct the experiments of the "Evaluation" chapter of my thesis. For the instructions in this file, we assume the SDS-tool has already been built, see the `README.md` file in the directory above.

The first three models of the section can be directly checked and compiled with the SDS-tool without much effort. See section "Simple Models" below.

For the more complex evaluations and performance measurements we employ python scripts, see the section "Complex Experiments".

Simple Models

Grading System

You can compile and run the "Grading System" model like this:

```
cd models/simple/grading_system
../../../../sdstool compile GradingSystem.abs GradingSystem.st
gen/erl/run
```

If you want to test out, how often the invocation order of methods is violated, if dynamic enforcement is not applied, please see the subsection `Grading System` in the `Complex Experiments` section of this file.

Heap Communication Example

You can compile and run the "Heap Communication" example model like this:

```
cd models/simple/responsive_ui
../../../../sdstool compile ResponsiveUI.abs ResponsiveUI.st
gen/erl/run
```

If you want to test out, how the model behaves, if the specified postconditions are violated, please see the subsection `Heap Communication Example` in the `Complex Experiments` section of this file.

Notification Service Example

You can compile and run the "Notification Service" example model like this:

```
cd models/simple/notification_service
../../../../sdstool compile NotificationService.abs NotificationService.st
gen/erl/run
```

Complex Experiments

We use python scripts to automatically alter ABS models and session types to conduct the more complex experiments presented in the thesis. We also use

these scripts to collect data about the performance of a model.

Therefore, the following dependencies need to be installed to run these experiments:

- Python 3.7
- pipenv 2018.11.26
- perf 5.3
- GNU time 1.9

For Arch Linux you can run the following command to install these dependencies:

```
sudo pacman -S python python-pipenv perf time
```

Next, all necessary python dependencies need to be installed by running

```
pipenv install
```

in the `evaluation` folder.

The experiments which conduct performance measurements will save the collected data as `.csv` files in the `cache` subfolder of the experiment. The content of these `.csv` files corresponds to the data tables in my thesis. The plots created from this data can be found in the same subfolder as `.pdf` files.

Grading System

If you run

```
pipenv run ./grading_system.py
```

the script will ask you, whether to enable the modifications necessary for dynamic enforcement or not. Next it will run the first example model of the thesis a 100 times and give statistics, how often the invocation order intended by the session type has been violated.

Heap Communication Example

If you run

```
pipenv run ./responsive_ui.py
```

the script will ask you, whether some modifications should be applied to the second example model of the thesis. Depending on the chosen modification, the postconditions specified in the session type are not fulfilled. Next, the script compiles and runs the model.

Performance Evaluation Example 1

The script `consecutive_calls.py` will conduct all experiments of phases I-III of the first performance evaluation experiment of the thesis:

```
pipenv run ./consecutive_calls.py
```

This script can take some time. The model template used for these experiments can be found in `models/complex/consecutive_calls`. To compute the `.csv` tables and `.pdf` plots mentioned above from the data collected in this experiment, please run the following command next:

```
pipenv run ./view_consecutive_calls.py
```

You can find the resulting files in `models/complex/consecutive_calls/cache`. The files are prefixed like this:

- Prefix of phase I: Method2DirectNoShuffle
- Prefix of phase II: Method2DirectNoShuffleAwait
- Prefix of phase III: Method2DirectReverseBusywait

If you want to run the experiments again, you have to delete this `cache` folder first:

```
rm -r models/complex/consecutive_calls/cache
```

Performance Evaluation Example 2

The script `notification_service_perf.py` will conduct all experiments of of the second performance evaluation experiment of the thesis:

```
pipenv run ./notification_service_perf.py
```

This script can take some time. The model template used for these experiments can be found in `models/complex/notification_service`. To compute the `.csv` tables and `.pdf` plots mentioned above from the data collected in this experiment, please run the following command next:

```
pipenv run ./view_notification_service_perf.py
```

You can find the resulting files in `models/complex/notification_service/cache`. If you want to run the experiments again, you have to delete this `cache` folder first:

```
rm -r models/complex/notification_service/cache
```