Agda

This folder contains developments written in Agda. In order to try out such programs, we offer a system to manage multiple Agda versions and libraries.

The quick way to get started with Agda is through the Docker images offered by Ting-gian Lua.

To make using them even easier, in this directory we offer an executable file called `run` which calls Docker and manages possible dependencies for you. The prerequisites are:

- Docker
- Python 3

Installing Docker and Python

Visit the Get Started with Docker web page and choose the appropriate option for your platform. In Windows, it might be necessary to activate the Hyper-V and Containers features. Take a look at the documentation.

Visit the Download Python web page and choose the appropriate option for your platform. It is important to use a 3.x (>= 3.6) version of Python, as 2.x is deprecated and will not work for this project.

Included projects

A few Agda projects are included in the `src/` folder.

- `Naturals.agda`. The proof of the commutativity of addition in the natural numbers.

  To try it, just execute `./run src/Naturals.agda` from the terminal.

- `Circle.agda`. The construction of the fundamental group of the circle.

  To try it, just execute `./run src/Circle.agda` from the terminal.

- `Fail.agda`. An example of an incorrect proof, so that the user can see the output when a program is not correctly typed.

  To try it, just execute `./run src/Fail.agda` from the terminal.

New projects

To type check another Agda project, follow these steps:

1. Create a file called `agda.json` at the root of the project, specifying the version of Agda to use and any necessary libraries. It must follow the specification defined in section Configuration file. Take a look at `src/agda.json` for an example.

2. Execute:

```
./run path/to/file.agda
```

The first run will take a bit longer because the script has to download the Docker images and libraries. After any Agda messages, the `run` script will output a success/failure message. If a success message is displayed, Agda has correctly type checked the program and hence its proof is deemed correct.

During execution, a folder named `.deps` will be created at the same level as the `agda.json` file. This folder can be safely removed at any time, although the following runs of the program will be faster if it is preserved.

Configuration file

In order to use different versions of Agda and different libraries, a configuration file can be set up. The file has to be named `agda.json` and must be placed at the root of the Agda project (i.e. at the folder that represents the root module).

The possible values for the file are as follows:

- `version`

  Description: the version of Agda to use.

  Type: string.

  Mandatory: yes.

  Examples: `"version": "2.5.3"`

- `libraries`

  Description: a list of objects representing the libraries to use with this project.

  Type: array.

  Mandatory: no.

  Each entry of the array must be an object with the following properties:

  - `repo`

    Description: the location of the Git repository that contains the library.

    Type: string (URI).

    Mandatory: yes.

    Examples: `"repo": "https://github.com/HoTT/HoTT-Agda"`

  - `libfile`

Description: the relative location inside the repository of the `*.agda-lib` file that represents the library.

Type: string.

Mandatory: yes.

Examples: `"libfile": "hott-core.agda-lib"`

- `branch`

  Description: the tag or branch of the repository to check out. If not included, by default the latest commit will be checked.

  Type: string.

  Mandatory: no.

  Examples: `"branch": "v0.1"`

For example, an Agda project using version 2.6.0.1 and the standard library, could use a configuration file like:

```
{
  "version": "2.6.0.1",
  "libraries": [
    {
      "repo": "https://github.com/agda/agda-stdlib",
      "branch": "v1.2",
      "libfile": "standard-library.agda-lib"
    }
  ]
}
```