— Sprint 1 —

Deadline: 24th of April

Preparation 1 - Module

Clone the git repository of your group. Create a new module in this repository using the script `writeModule.py`, whose only argument is the new modules name. You can execute this script with the following command in the SMT-RAT root directory:

```
python writeModules.py <name>
```

Now you should find a new folder named `<name>Module` in the folder `src/smtrat-modules/`.

Within this module, you will work for the rest of the practical course. Information about the members of a module, the interfaces to implement and auxiliary functions are available in the SMT-RAT Wiki [1].

Preparation 2 - Strategy

Next you need a new strategy to actually use your module. Therefore you create a new file in the folder `src/smtrat-strategies/strategies/`, you may copy one of the existing files e.g. `RatOne.h`. Change the filename to the desired name for your strategy.

Now you have to modify the content of your strategy file. The class (and constructor) have to be named identical to the filename. Also change the includes to `smtrat-solver/Manager.h` and your module from `smtrat-modules/<name>Module/<name>Module.h`. Last you have to adapt the actual strategy to only use your module.

To change the used strategy to yours, execute

```
ccmake ../
```

and change the option `SMTRAT_Strategy` to the name of your strategy. If you compile SMT-RAT now, the current version of your module will be used so you can check whether it compiles.

Main task

We are the technology leader for VaaS[2] with our revolutionary WolVeriNe[3] service. In a nutshell, we employ a SCAM[4] architecture to execute verification tasks on USER[5] computers and collect the results in a private stateless blockchain. For the verification task itself we heavily rely on a diverse set of solving backends.

We hired you to implement an SMT solver arguing about uninterpreted domains that we want to integrate into WolVeriNe. Starting with a few basic tasks, we'll figure out possible applications for this tool along the way.

To get started we'd like you to solve the set of toy examples provided at `https://ths.rwth-aachen.de/teaching/ss19/swp-smt/task1.zip`

They all consist of a collection of variables – their meaning is not relevant for you and we have anonymized their names – and some equalities and disequalities. Just figure out whether the resulting problem is satisfiable. And if there is a solution, just tell us which variables are equal.

[1] `https://github.com/smtrat/smtrat/wiki/4---Implementing-Further-Modules`
[2] Verification-as-a-Service
[3] Worldwide Verification Network
[4] Serverless Computing and Allocation Model
[5] Unaware Self-sacrificial External Resource