Prof. Dr. Rüdiger Ehlers TU Clausthal

Model Checking and Games 2019 – Assignment Sheet 8

Due: Thursday, 12th September 2019, before the lecture

Please indicate your **name**. You can work in **groups** of up to **three** students. Only one submission per group is necessary. However, in the tutorials every group member must be able to present the solutions to each problem solved by your group.

Please submit your solutions by e-mail to `ruediger.ehlers@tu-clausthal.de` or hand them in in paper form right before the lecture.

Note that you will need 50% of the points on all exercise sheets in order to take the exam. You may be asked to present your solutions in the tutorial, especially if you work in a group. We aim for asking everyone taking part in the course to present at least once during the block course.

Exercise 0: Preparing the use of a SAT solver

Before you can solve the following exercise, you need to get a SAT solver to run. **There are two choices** described in the following.

Picosat on Linux

For example, the solver `picosat` can be downloaded and compiled on these machines using the following commands (tested on the machine `as.rz.tu-clausthal.de`):

```
cd /tmp
mkdir <somePathName>
cd <somePathName>
wget http://fmv.jku.at/picosat/picosat-960.tar.gz
tar -xvzf picosat-960.tar.gz
cd picosat-960
./configure
make
```

This will create the executable `picosat`, which can then be copied elsewhere. For example, the following command will copy it to your local home folder:

```
cp picosat ~/
```

You can then edit CNF text files with an editor (such as `nano`) and run the solver using the following commands (assuming that you copied the solver to your home directory):

```
~/picosat <inputFile>
```

Cryptominisat in a Web Browser

As an alternative, you can also use the SAT solver `CryptoMiniSat` in a browser:
`https://www.msoos.org/2013/09/minisat-in-your-browser/` -

The output of the solver is a bit different in this case, though.

Exercise 1: Satisfiability Solving (30 pts.)

Obtain some practical experience with using a Satisfiability Solver.

Translate the following Boolean formula into conjunctive normal form and solve it using a SAT solver:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee (x_3 \wedge \neg x_2))$$

Exercise 2: Satisfiability Solving (35 pts.)

In the example for Algorithm 1 in the lecture, we fixed a variable ordering for the choices that the algorithm makes. Can you construct a small SAT instance that is solved quickly by Algorithm 1 ("A first, very simple SAT solving algorithm") for one ordering, but takes much longer for another ordering?

You can either use the number of conflicts or the number of nodes in the trees that depict the runs of Algorithm 1 as measure for the running time of the algorithm. In any case, justify your answer.

Exercise 3: Satisfiability Solving (35 pts.)

Apply the DPLL algorithm on the following SAT instance:

$$\psi = (\neg x_1 \vee x_2 \vee \neg x_5) \wedge (x_2 \vee \neg x_4) \wedge x_1 \wedge (x_4 \vee x_5) \wedge (\neg x_4 \vee \neg x_5)$$

Show the steps performed and provide a satisfying assignment to the variables $x_1, \ldots, x_5$.