Introduction

Rebeca model-checking tool consists of two components: a translator and a model-checking engine. The Rebeca translator converts the input Rebeca model to some C++ files. The model-checking files are added to the converted files. Package of these C++ files contains all the required files for model checking called model-checking engine.

Model checking engine is compiled to an executable file. Running the executable file result in applying model checking algorithm on the given Rebeca model and reporting the verification result.

How to generate C++ files

All the required libraries for generating C++ files are included in a Java executable Jar file "org.rebecalang.rmc-2.3.0-SNAPSHOT.jar". You can run this file with JRE 1.6 or upper using the following parameters.

Parameter	Example	Description
-s, --source	-s /home/test/model.rebeca -s myModel.rebeca	Location of Rebeca source file
-p, --property	-p /home/test/model.property -p myModel.property	Location of Rebeca model property file
-o, --output	-o /home/test/modelFolder -o myModelFolder	Target of generated C++ file
-v, --version	-v 2.0 -v 2.1	Compiler version which could be Rebeca 2.0 or Rebeca 2.1. Features of Timed Rebeca, Probabilistic Rebeca, and Timed Probabilistic Rebeca are enabled in version 2.1.
-x, --exporttransitionsystem	-x	Exporting the state space in XML format in "statespace.xml" file. The exported state space can be visualized using "state space analysis" library.
-e, --extension	-e CoreRebeca -e TimedRebeca -e ProbabilisticRebeca -e TimedProbabilistiRebeca	Specifying the type of Rebeca model
-h	-h	Print the parameters description

Finally two examples of typical commands for model checking Rebeca and Timed Rebeca models are depicted at the following.

- java -jar org.rebecalang.rmc-2.3.0-SNAPSHOT.jar -s model.rebeca -p model.property -o rmc
- java -jar org.rebecalang.rmc-2.3.0-SNAPSHOT.jar -s timed-model.rebeca -p timed-model.property -o rmc -v 2.1 -e TimedRebeca -x

How to execute the generated C++ files

The result C++ files can be compiled using any distribution of C++ compilers. In the following example we use g++ to compile the generated C++ files and set the compilation output file to “executable”.

```
g++ *.cpp -w -o executable
```

Running “executable” file results in model checking of the model.