Linnéuniversitetet
Kalmar Växjö

Test Suite
ActivFORMS Virtual Machine

Author: M. Usman Iftikhar,
Danny Weyns
KU Leuven, Belgium
Linnaeus University, Sweden
Email: usman.iftikhar@lnu.se

Linnéuniversitetet
Kalmar Växjö

ActivFORMS

ActivFORMS (Active FORmal Models for Self-adaptation) is a formally founded approach that aims to provide guarantees for the correct execution of MAPE-K feedback loops of self-adaptive systems [1]. The key component of ActivFORMS is a virtual machine that executes formal models of a feedback loop that are specified as a network of timed automata. To provide evidence that the virtual machine works correctly, we created a test suite, which tests the correct execution of models by the virtual machine. The test suite comprises models of the different model primitives used. To check the correctness of a model primitive, we compare the results of the execution of the model of the model primitive by the Uppaal verifier [2] with the results generated by the execution with the ActivFORMS virtual machine. Concretely, we exploit the simulation feature of Uppaal that allows saving execution traces of model executions. The test suite matches the results of the execution of model execution by the virtual machine with the execution traces.

Model Primitives

We use the following tests to verify the execution of model primitives:

1. VariableInitialization.xml: Verify initialization of the variables with their initial values.
2. ProcessInitialization.xml: Tests process initialization and instantiation.
3. Loops.xml: Tests loops supported by Uppaal such as while, do while and for loop, iterative and inner loops.
4. Functions.xml: Tests template functions, global and system declarations. Furthermore, tests function calls with parameters (passed by value and by reference).
5. Signals.xml: Tests type of signals, i.e., binary, broadcast, urgent, urgent broadcast signals, and also, tests array of signals.
6. Time.xml: Checks time constraints include invariants and guards.
7. Locations.xml: Checks committed and urgent locations with their priority.
8. Structures.xml: Checks structures, their initialization, and uses.
9. Arrays.xml: Test arrays, initialization, in loops, etc.
10. Operators.xml: Checks unary, binary and ternary operators with different data operands and expressions.

Running the Test Suite

To execute the test suite, double click the provided jar file or execute the following command via the console:

```
java –jar TestSuite
```

The default execution of the test suite (without any parameters) reads the models and trace files from the "*tests*" folder in the test suite path. To change the tests folder path or to execute a single test, use the following commands:

- -d directoyPath: This command parameter changes the tests folder directory.
- -f modelfilePath tracefilePath: This command executes only one test by using the paths of the model and trace files.
- -t timeunit: This option enables changing the model time (units in milliseconds). The default time unit is 10 milliseconds.
- -dt maxDelayTransitions: This option forces the virtual machine to take a transition after a maximum number of delay transitions. A delay transition only passes time without an actual transition from one automata location to other. The default value for this option is maximum 10 delay transitions. If this option is set to -1, the virtual machine turnoff the option. If this option is set to 0, the virtual machine will not perform any delay transition, and, if possible, immediately takes an enabled transition.

Adding New Tests

New tests can be added dynamically to the test suite. To that end, the Uppaal Timed Automata Parser Library needs to be installed. Information about installing the Parser library can be found here: http://www.pi-identity.com/blog/2011/10/26/using-uppaal-for-trace-interpretation/

After installing the parser library, there are four steps to create a trace file from the model.

1. Save the trace file from the Uppaal simulator. This file is an unreadable binary. We need to make this trace file readable for the test suite.

2. Compile the model using the following command:

```
UPPAAL_COMPILE_ONLY=1 verifyta model.xml - > model.if
```

   The Verifyta utility can be found in the Uppaal folder.

3. After compiling the model, use both the compiled model and trace file to create a readable trace file.

```
tracer model.if trace.xtr > trace.txt
```

   The tracer utility can be found in the installed automata parser library.

4. Save the model and the trace files in the tests folder of the test suite. To add new tests, the model and trace files should be saved with the same name and with ".xml" and ".txt" extensions.

References:

[1]. M. U. Iftikhar, D. Weyns, ActivFORMS: Active FORmal Models for Self-adaptation, 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2014, Hyderabad, India

[2]. G. Behrmann, R. David, and K. G. Larsen. A tutorial on Uppaal (pages 200–236). Springer, 2004.