Institute for System Programming of the Russian Academy of Sciences

MicroTESK User Guide (UNDER DEVELOPMENT)

Moscow 2016

Contents

Chapter 1

Installation

1.1 System Requirements

MicroTESK is a set of Java-based utilities that are run from the command line. It can be used on ***Windows***, ***Linux*** and ***OS X*** machines that have ***JDK 1.7 or later*** installed. To build MicroTESK from source code or to build the generated Java models, ***Apache Ant version 1.8 or later*** is required. To generate test data based on constraints, MicroTESK needs the ***Microsoft Research Z3*** or ***CVC4*** solver that can work under the corresponding operating system.

1.2 Running MicroTESK

To generate a Java model of a microprocessor from its nML specification, a user needs to run the compile.sh script (Unix, Linux, OS X) or the compile.bat script (Windows). For example, the following command generates a model for the miniMIPS specification:

```
sh bin/compile.sh arch/minimips/model/minimips.nml
```

NOTE: Models for all demo specifications are already built and included in the MicroTESK distribution package. So a user can start working with MicroTESK from generating test programs for these models.

To generate a test program, a user needs to use the generate.sh script (Unix, Linux, OS X) or the generate.bat script (Windows). The scripts require the following parameters:

1. model name

2. test template file

3. target test program source code file

For example, the command below runs the euclid.rb test template for the miniMIPS model generated by the command from the previous example and saves the generated test program to an assembler file. The file name is based on values of the –code-file-prefix and –code-file-extension options.

```
sh bin/generate.sh minimips arch/minimips/templates/euclid.rb
```

To specify whether Z3 or CVC4 should be used to solve constraints, a user needs to specify the -s or –solver command-line option as z3 or cvc4 respectively. By default, Z3 will be used. Here is an example:

```
sh bin/generate.sh -s cvc4 minimips arch/minimips/templates/constraint.rb
```

More information on command-line options can be found on the Command-Line Options section.

1.3 Command-Line Options

MicroTESK works in two modes: specification translation and test generation, which are enabled with the –translate (used by default) and –generate keys correspondingly. In addition, the –help key prints information on the command-line format.

The –translate and –generate keys are inserted into the command-line by compile.sh/compile.bat and generate.sh/generate.bat scripts correspondingly. Other options should be specified explicitly to customize the behavior of MicroTESK. Here is the list of options:

Full name	Short name	Description	Requires
–help	-h	Shows help message	
–verbose	-v	Enables printing diagnostic messages	
–translate	-t	Translates formal specifications	
–generate	-g	Generates test programs	
–output-dir <arg>	-od	Sets where to place generated files	
–include <arg>	-i	Sets include files directories	–translate
–extension-dir <arg>	-ed	Sets directory that stores user-defined Java code	–translate
–random-seed <arg>	-rs	Sets seed for randomizer	–generate
–solver <arg>	-s	Sets constraint solver engine to be used	–generate
–branch-exec-limit <arg>	-bel	Sets the limit on control transfers to detect endless loops	–generate
–solver-debug	-sd	Enables debug mode for SMT solvers	–generate
–tarmac-log	-tl	Saves simulator log in Tarmac format	–generate
–self-checks	-sc	Inserts self-checking code into test programs	–generate
–arch-dirs <arg>	-ad	Home directories for tested architectures	–generate
–rate-limit <arg>	-rl	Generation rate limit, causes error when broken	–generate
–code-file-extension <arg>	-cfe	The output file extension	–generate
–code-file-prefix <arg>	-cfp	The output file prefix (file names are as follows prefix_xxxx.ext, where xxxx is a 4-digit decimal number)	–generate
–data-file-extension <arg>	-dfe	The data file extension	–generate
–data-file-prefix	-dfp	The data file prefix	–generate

1.4 Overview

Chapter 2

Appendixes

2.1 References

Bibliography

[1] M. Freericks. *The nML Machine Description Formalism.* Technical Report TR SM-IMP/DIST/08, TU Berlin CS Department, 1993.