Institute for System Programming of the Russian Academy of Sciences

MicroTESK Installation and Usage (UNDER DEVELOPMENT)

Moscow 2017

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 on the corresponding operating system.

1.2 Installation Steps

To install MicroTESK, the following steps should be performed:

1. Download from `http://forge.ispras.ru/projects/microtesk/files` and unpack the MicroTESK installation package (the `.tar.gz` file, latest release) to your computer. The folder to which it was unpacked will be further referred to as the installation directory (`<installation dir>`).

2. Declare the `MICROTESK_HOME` environment variable and set its value to the path to the installation directory (see the Setting Environment Variables section).

3. Set the `<installation dir>/bin` folder as the working directory (add the path to the `PATH` environment variable) to be able to run MicroTESK utilities from any path.

4. Note: Required for constraint-based generation. Download and install constraint solver tools to the `<installation dir>/tools` directory (see the Installing Constraint Solvers section).

1.2.1 Setting Environment Variables

Windows

1. Open the `System Properties` window.

2. Switch to the `Advanced` tab.
3. Click on `Environment Variables`.
4. Click `New..` under `System Variables`.
5. In the `New System Variable` dialog, specify variable name as `MICROTESK_HOME` and variable value as `<installation dir>`.
6. Click `OK` on all open windows.
7. Reopen the command prompt window.

Linux and OS X

Add the command below to the `~.bash_profile` file (***Linux***) or the `~/.profile` file (***OS X***):

```
export MICROTESK_HOME=<installation dir>
```

To start editing the file, type `vi ~/.bash_profile` (or `vi ~/.profile`). Changes will be applied after restarting the command-line terminal or reboot. You can also run the command in your command-line terminal to make temporary changes.

1.2.2 Installing Constraint Solvers

To generate test data based on constraints, MicroTESK requires external constraint solvers. The current version supports the Z3 and CVC4 constraint solvers. Constraint executables should be downloaded and placed to the `<installation dir>/tools` directory.

Using Environment Variables

If solvers are already installed in another directory, to let MicroTESK find them, the following environment variables can be used: `Z3_PATH` and `CVC4_PATH`. They specify the paths to the Z3 and CVC4 excutables correspondingly.

Installing Z3

- ***Windows*** users should download Z3 (32 or 64-bit version) from the following page:`http://z3.codeplex.com/releases` and unpack the archive to the `<installation dir>/tools/z3/windows` directory. Note: the executable file path is `<windows>/z3/bin/z3.exe`.
- ***UNIX*** and ***Linux*** users should use one of the links below and and unpack the archive to the `<installation dir>/tools/z3/unix` directory. Note: the executable file path is `<unix>/z3/bin/z3`.

Debian x64	`http://z3.codeplex.com/releases/view/101916`
Ubuntu x86	`http://z3.codeplex.com/releases/view/101913`
Ubuntu x64	`http://z3.codeplex.com/releases/view/101911`
FreeBSD x64	`http://z3.codeplex.com/releases/view/101907`

- ***OS X*** users should download Z3 from `http://z3.codeplex.com/releases/view/101918` and unpack the archive to the `<installation dir>/z3/osx` directory. Note: the executable file path is `<osx>/z3/bin/z3`.

Installing CVC4

- ***Windows*** users should download the latest version of CVC4 binary from `http://cvc4.cs.nyu.edu/builds/win32-opt/` and save it to the `<installation dir>/tools/cvc4/windows` directory as `cvc4.exe`.
- ***Linux*** users download the latest version of CVC4 binary from `http://cvc4.cs.nyu.edu/builds/i386-linux-opt/unstable/` (32-bit version) or `http://cvc4.cs.nyu.edu/builds/x86_64-linux-opt/unstable/` (64-bit version) and save it to the `<installation dir>/tools/cvc4/unix` directory as `cvc4`.
- ***OS X*** users should download the latest version of CVC4 distribution package from `http://cvc4.cs.nyu.edu/builds/macos/` and install it. The CVC4 binary should be copied to `<installation dir>/tools/cvc4/osx` as `cvc4` or linked to this file name via a symbolic link.

1.3 Installation Directory Structure

The MicroTESK installation directory contains the following subdirectories:

arch	Microprocessor specifications and test templates
bin	Scripts to run modeling and test generation tasks
doc	Documentation
etc	Configuration files
gen	Generated code of microprocessor models
lib	JAR files and Ruby scripts to perform modeling and test generation tasks
src	Source code of MicroTESK

1.4 Running

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:

- model name;
- test template file;
- 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.

1.5 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
–default-test-data	-dtd	Enables generation of default test data	–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 <arg>	-dfp	The data file prefix	–generate

–exception-file-prefix <arg>	-efp	The exception handler file prefix	–generate
–program-length-limit <arg>	-pll	The maximum number of instructions in output programs	–generate
–trace-length-limit <arg>	-tll	The maximum length of execution traces of output programs	–generate
–comments-enabled	-ce	Enables printing comments; if not specified no comments are printed	–generate
–comments-debug	-cd	Enables printing detailed comments; must be used together with –comments-enabled	–generate
–no-simulation	-ns	Disables simulation of generated test programs on the model	–generate
–time-statistics	-ts	Enables printing time statistics	–generate

1.6 Settings File

Default values of options are stored in the `<MICROTESK_HOME>/etc/settings.xml` configururation file that has the following format:

```
<?xml version="1.0" encoding="utf-8"?>
<settings>
  <setting name="random-seed" value="0"/>
  <setting name="branch-exec-limit" value="1000"/>
  <setting name="code-file-extension" value="asm"/>
  <setting name="code-file-prefix" value="test"/>
  <setting name="data-file-extension" value="dat"/>
  <setting name="data-file-prefix" value="test"/>
  <setting name="exception-file-prefix" value="test_except"/>
  <setting name="program-length-limit" value="1000"/>
  <setting name="trace-length-limit" value="1000"/>
  <setting name="comments-enabled" value=""/>
  <setting name="comments-debug" value=""/>
  <setting name="default-test-data" value=""/>
  <setting
    name="arch-dirs"
    value="cpu=arch/demo/cpu/settings.xml:minimips=arch/minimips/settings.xml"
  />
</settings>
```