RiTHM Manual

Table of Contents

[1.Overview 1](#__RefHeading__613_755760951)

[2.Prerequisites before the installation 1](#__RefHeading__615_755760951)

[apt-get Packages 1](#__RefHeading__617_755760951)

[External 2](#__RefHeading__619_755760951)

[Other Dependencies 2](#__RefHeading__621_755760951)

[3.Installation 3](#__RefHeading__623_755760951)

[4.Using the tool 3](#__RefHeading__625_755760951)

[Running from GUI 3](#__RefHeading__627_755760951)

[Running from Command-Line 4](#__RefHeading__629_755760951)

[5.Configuration File 4](#__RefHeading__631_755760951)

# Overview

The tool RiTHM (Runtime Time-triggered Heterogeneous Monitoring) takes a C program under inspection and a set of LTL properties as input and generates an instrumented C program that is verified at run time by a time-triggered monitor. The output C program can be instrumented in two ways: it either **self-monitors** itself (i.e., the monitoring code is weaved with the input program), or it incorporates an **external monitor** thread. In both cases, the monitor is invoked in a time-triggered fashion, ensuring that the states of the program can be reconstructed at each invocation by using efficient instrumentation. The monitor verification decision procedure is sound and complete and takes advantage of the GPU many-core technology to speedup monitoring and reduces the runtime overhead.

# Prerequisites before the installation

The tool has been tested on Ubuntu 11.10 32-bit OS.

### apt-get Packages

* ia32-libs: if the user is running Ubuntu 64-bit, i132-libs package should be installed. This package allows the user to skip compilation process specifically for 64-bit machine and run 32-bit executables on 64-bit platform
* realpath: this package used to resolve path parsing issues and path dependencies; used in some of the invocation scripts.
* qmake: if the user wishes to build the GUI from the committed source files

### External

The tool allows the user to choose whether the verification is performed on CPU or on GPU. Thus, the OpenCL supporting packages should be installed previously.

* For system with AMD/ATI GPU: AMD GPU OpenCL SDK can be downloaded from here:
* <http://developer.amd.com/tools/hc/AMDAPPSDK/downloads/Pages/default.aspx>
* For systems with NVIDIA GPU: NVIDI(A OpenCL SDK can be downloaded from here:
* <http://www.nvidia.com/Download/index.aspx?lang=en-us>

Remark: if system uses AMD GPU/APU card, but AMD GPU OpenCL SDK is not installed, it will be installed automatically when running ./build\_deps.h

### **Other Dependencies**

The following packages will be packaged along with the RiTHM source code and will be installed automatically when running ./build\_deps.sh:

* yices
* lp\_solve
* libconfig
* open CSV (java lib)
* apache commons (java lib)
* LLVM
* clang

# Installation

Tool install the RiTHM please follow the installation steps:

1. Install the external dependencies
2. Download the tar ball (containing all the binaries) and extract it somewhere on the local machine. The tar is available at

<https://bitbucket.org/embedded_software_group/rvtool/src>

1. Run installation script ./build\_deps.sh from the root directory. The installation script can run without root access (no need in “sudo”).
2. Run make form rppt directory (no need in “sudo”)
3. Run the tool by executing the file in <tool top dir>/build/rvtool

# Using the tool

RiTHM can be run both from GUI environment (recommended) and from command-line. The GUI environment allows specifying all the available parameters; then it wraps the parameters and invokes appropriate script that, in turn, invokes the scripts with the corresponding command-line parameters.

### Running from GUI

Following is the snapshot of the main GUI window and explanation of the input fields:

* “Source directory”

Specify directory containing the source files of the application under inspection.

* “Property filepath”

Specify the path to the configuration file (see Section 5).

* “Algorithm type”

Specify the verification algorithm type. Currently we have developed four different algorithms for the properties verification. They can be listed in the declining order of the CPU engagement: Sequential, Partial Offload, Finite-History and Infinite-History algorithm.

* “Invocation type”

Specify the blocking type of the verification process. The verification process might be either blocking (main thread invokes the flush function and waits until it’s done) or non-blocking (main threads designates a worker thread from the pool to perform the verification).

* “Target sampling period”

This option controls the target sampling period.

* “Buffer size”

By default, the buffer of the program states is flushed by the instrumented code. This option allows the user to specify different behaviour: flush the verification buffer once it is full.

* “External, Self”

Specify the monitoring mode and method to identify the instrumentation points.

### Running from Command-Line

The example of command-line invocation:

./run.sh ../test/Fibo ../test/out ../test/Fibo/fibo.cfg ilp 1 \_GOOMF\_enum\_alg\_seq \_GOOMF\_enum\_sync\_invocation 100

Where

* run.sh is the main script;
* ../test/Fibo is the source directory
* ../test/out is the output directory for the instrumented files
* ../test/Fibo/fibo.cfg is the path to the configuration file
* ilp is the type of the heuristic
* 1 is the target sampling period
* GOOMF\_enum\_alg\_seq means that the verification algorithm is sequential and runs on the CPU
* GOOMF\_enum\_sync\_invokation means that the verification is synchroneous
* 100 is the size of the verification buffer

# Configuration File

Configuration file should be provided by the developer and should contain the list of the monitored properties along with the mapping of the mapping of the properties predicates to the local and global variables in the inspected C program.

Example of the configuration file:

//A configuration file that describes monitoring objects

//name of the process under scrutiny

program\_name = "QLogAudit";

//developer can specify his own verification functions

functions = (

"bool IMUSanityCheckPhi(float phi1, float teta1, float phi2, float p1, float q1, float r1, float delta)

{

float temp = ((phi2 - phi1) / 0.01) - (p1 + q1\*sin(phi1)\*tan(teta1) + r1\*cos(phi1)\*tan(teta1));

return (temp < delta && temp > -delta);

}",

"bool xisNegative(x)

{

return (x < 0);

}")

//LTL properties specified in Future-LTL syntax or in AT&T FSM format

properties = (

{name = "prop1"; formalism="LTL"; syntax = "[] (a && b)"},

{name = "prop2"; formalism="LTL"; syntax = "[] (a U b)"},

{name = "prop3"; formalism="LTL"; syntax = "[] (c -> X (d U ! c))"},

{name = "satellites"; formalism="FSM"; syntax =

"digraph G {

\"(0, 0)\" -> \"(1, 1)\" [label = \"(e&&f)\", action = \"bind(timestamp)\"];

\"(0, 0)\" -> \"(1, 1)\" [label = \"(e)\", action = \"bind(timestamp)\"];

\"(0, 0)\" -> \"(0, 0)\" [label = \"(f)\"];

\"(0, 0)\" -> \"(0, 0)\" [label = \"(<empty>)\"];

\"(1, 1)\" -> \"(1, 1)\" [label = \"(e&&f)\"];

\"(1, 1)\" -> \"(-1, 2)\" [label = \"(e)\"];

\"(1, 1)\" -> \"(0, 0)\" [label = \"(f)\"];

\"(1, 1)\" -> \"(0, 0)\" [label = \"(<empty>)\"];

\"(-1, 2)\" -> \"(-1, 2)\" [label = \"(e&&f)\"];

\"(-1, 2)\" -> \"(-1, 2)\" [label = \"(e)\"];

\"(-1, 2)\" -> \"(-1, 2)\" [label = \"(f)\"];

\"(-1, 2)\" -> \"(-1, 2)\" [label = \"(<empty>)\"];

\"(-1, 2)\" [label=\"(-1, 2)\", style=filled, color=red]

\"(1, 1)\" [label=\"(1, 1)\", style=filled, color=pink]

\"(0, 0)\" [label=\"(0, 0)\", style=filled, color=darkseagreen1]

}"

} );

//mapping of the predicates on the program variables

predicates = ( {name = "a"; syntax = "x > y"},

{name = "b"; syntax = "z < 0"},

{name = "c"; syntax = "xisNegative(float x)"},

{name = "d"; syntax = "z == 0"},

{name = "e"; syntax = "num\_of\_sats < 3"},

{name = "f"; syntax = "timestamp < param1 + 10"});

//all program variables being involved in the monitoring and their types

program\_variables = ( {name = "x"; type = "float"},

{name = "y", type = "float"},

{name = "z", type = "float"},

{name = "num\_of\_sats", type = "unsigned int"},

{name = "timestamp", type = "unsigned long"});