Model Checking Contest 2013

Submission Manual

http://mcc.lip6.fr

Contents

1 Introduction

This document presents the submission procedure of the Model Checking Contest @ Petri Nets 2013. Prior to any submission, please check that you are in the conditions of the Model Checking Contest @ Petri Nets 2013. These rules are available at `http://mcc.lip6.fr/rules.php`.

Please contact `Fabrice.Kordon@lip6.fr` if you find any inconsistency or problem in this document or in the submission procedure.

IMPORTANT: This year, the process has been refined and simplified from the tool developer's point of your. To do so, we applied several suggestions raised during the discussion when results from the MCC'2012 were presented in Hamburg.

The main improvement is the development of a simple and separate benchmark environment: ***B*enchKit**. The new scripts rely on ***B*enchKit** for the execution of tools, the extraction and management of Memory/CPU information and to confine the execution. Execution on clusters is also handled by ***B*enchKit**.

B*enchKit**[1] is developed within the context of the ***CosyVerif[2] project, supported by the **MeFoSyLoMa** group[3].

The 2013 Submission Kit. The submission kit is composed of the following elements:

- The public release of ***B*enchKit** (`http://cosyverif.org/BENCHMARKS/BenchKit-distrib.tgz`); to operate it, please report to the user manual (`http://cosyverif.org/BENCHMARKS/BenchKit.pdf`),
- A disk image preinstalled with models, formulas, and a dummy tool allowing tool developers to see how the system works (`http://cosyverif.org/BENCHMARKS/BenchKit-mcc-disk_image.vdmk`),
- the public `ssh` key associated with the main account to be used in the virtual machine. This disk image runs under Linux, if you need another distribution or another operating system, please have a look on section 3.

This disk image will be operated as a QEMU/KVM virtual machine. The dis image has been designed t be compatible with other virtualization environment (it was tested under **VirtualBox**[4]).

2 Integrating your Tool in the Submission Kit

This sections explains to you how to let you tool interact with the submission kit for the MCC'2013.

***B*enchKit** and the provided disk image are ready to be used together, provided that you adapt the description of the machines to be used.

2.1 Content of the Disk Image

Th disk image is a Linux system (Debian 7.0 in 32 bit mode) including two accounts: `root` and `mcc`. for these two logins, a public-key associated to the private key provided to you is installed. You may thus connect directly using this key. You may add more keys but never remove this one since it will be used to operate your tool during the evaluation phase.

The home directory of the *mcc* is structured as presented in figure 1. You find a unique directory, `BenchKit`, containing:

- `bin`, a directory where you should put all the executable and libraries of your tool,

[1] `http://BenchKit.CosyVerif.org`
[2] `http://CosyVerif.org`
[3] `http://www.mefosyloma.fr`
[4] `http://www.virtualbox.org`

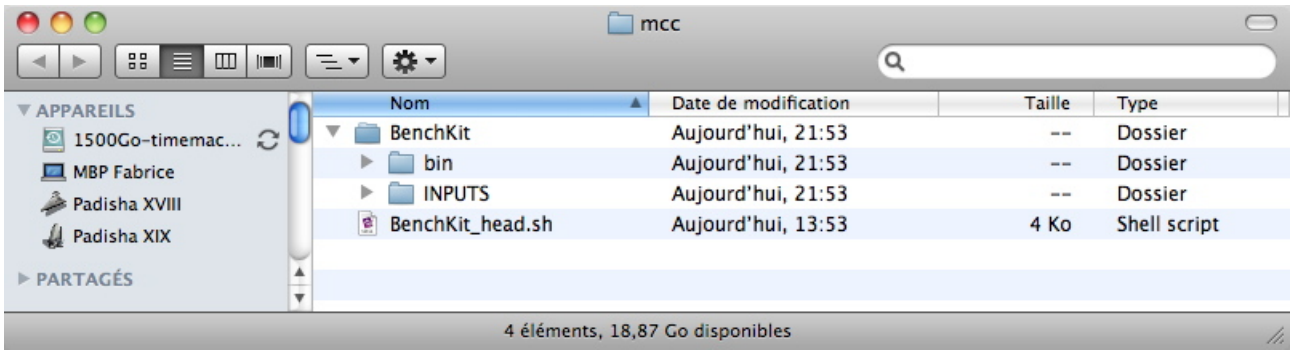

Figure 1: Structure of `mcc` user home directory in the virtual image.

- `INPUTS`, a directory that contains all the models provided to you. This directory contains one directory per input to be evaluated. Each directory contains a fixed number of files (PNML, properties, etc) that are detailed in section 2.3.
- `BenchKit_head.sh`, a script executed from outside the virtual machine ; it is dedicated to the invocation of your tool.

2.2 Overview of the Execution Procedure

The MCC'2013 execution procedure mainly relies on ***B*ench*K*it**. Execution of your tool is driven by the script `BenchKit_head.sh` that is executed remotely on the virtual machine.

So, for each examination (state space generation, evaluation of properties, etc.), and each model instance, ***B*ench*K*it**:

- starts the virtual machine,
- connects to the virtual machine as `root` to operate CPU and memory monitoring,
- connects to the virtual machine as `mcc` to operate your tool for the examination on the model instance,
- stops the virtual machine[5] after retrieving the observed data and add them into a CSV file.

Please note that all tools are operated in the same conditions to avoid any deviation in the measurement of CPU and memory. Among the possible machines that will run the VMs, we envision:

- a 23-nodes cluster equipped with PowerEdge R410 (6 ports gigabits) and 1.5To local disks. Each node operated an Intel Xeon E5645@2.40GHz (6 cores, 12 threads) and had 8GB of memory (DDR3, 1333),
- a 24-core Xeon 2.40GHz processor with 128GByte of memory.

2.3 Connecting Your Tool to the Execution Script

For the integration of your tool in ***B*ench*K*it**, please refer to the corresponding user manual. Please note that the mechanism has been considerably simplified since last year. However, it is based on the same principle so, its adaptation for those who already participated in previous years should not be a problem.

Basically, the only file you have to adapt is `BenchKit_head.sh`. This script operated by ***B*ench*K*it** in a directory that contains all you need to perform the current examination. This directory contains several files:

[5] the `halt` command will be invoked from the root account, either if your tool terminates or if it timeouts

- `model.pnml`, that is the initial PNML file that you may use for the model checking contest,
- `iscolored`, that contains either FALSE (if this is a colored model) or TRUE (if it is a P/T one),
- ⟨`category`⟩`.prop` and ⟨`category`⟩`.xml` where ⟨`category`⟩ is the value of the `BK_EXAMINATION` environment variable when it designate an examination where properties are required; `.prop` files contain a list of properties in textual format while `.xml` files are the corresponding properties stored as an XML tree (see documentation about properties),

  Since some properties may not be computed (see the note below the table of examinations), some of these files may be missing, then meaning that the examination does not exist for the corresponding model.
- `equiv_col` (for P/T nets) or `equiv_pt` (for Colored nets), a file containing FALSE (there is no equivalent colored or P/T net) or TRUE (there is an equivalent colored of P/T net),
- `instance`, a file containing the value of the current instance for the model.

When preparing your virtual machine, you may add dedicated files (e.g. adapted description of the model for your tool). **You are not allowed to cache pre-computed results.**

IMPORTANT: There will be "surprise models" meant to evaluate tools in a default mode (*i.e.* no dedicated optimization). These will not be present in the initial archive provided to you but an empty file named `NewModel` will be located in the directory too. **For these models, only a PNML description will be provided and thus, to participate in this category, your tool will be required to import the PNML format.**

You now just have two environment variables that are transmitted to you by the system:

- `BK_EXAMINATION`, that specifies the examination we expect your tool to perform. Possible values for this variable are:

Value	Signification
`StateSpace`	we ask for state space generation only
`ReachabilityDeadlock`	we evaluate reachability properties dealing with transition deadlocks only
`ReachabilityFireability`	we evaluate reachability properties dealing with transition fireability only
`ReachabilityPlaceComparison`	we evaluate reachability properties dealing with the comparison of places marking only
`ReachabilityCardinalityComparison`	we evaluate reachability properties dealing with checking cardinality of marking only
`ReachabilityMarkingComparison`	we evaluate reachability properties dealing with marking comparison only
`ReachabilityMix`	we evaluate reachability properties dealing with all the previous type of atomic proposition
`CTLFireability`	we evaluate CTL properties dealing with transition fireability only
`CTLPlaceComparison`	we evaluate CTL properties dealing with the comparison of places marking only
`CTLCardinalityComparison`	we evaluate CTL properties dealing with checking cardinality of marking only
`CTLMarkingComparison`	we evaluate CTL properties dealing with marking comparison only
`CTLMix`	we evaluate CTL properties dealing with all the previous type of atomic proposition
`LTLFireability`	we evaluate LTL properties dealing with transition fireability only
`LTLPlaceComparison`	we evaluate LTL properties dealing with the comparison of places marking only
`LTLCardinalityComparison`	we evaluate LTL properties dealing with checking cardinality of marking only
`LTLMarkingComparison`	we evaluate LTL properties dealing with marking comparison only
`LTLMix`	we evaluate LTL properties dealing with all the previous type of atomic proposition

IMPORTANT note 1: marking comparison works only for colored nets and their P/T equivalents. So, purely P/T models do not support this examination (since comparison is done on multisets of color, that has no signification for purely P/T Nets). So, the following models will never be invoked for this examination: Dekker, Echo, Eratosthenes, FMS, Kanban, MAPK, Planning, Railroad, RessAllocation, Ring

IMPORTANT note 2: since the TokenRing only contains one single place, no place comparison (marking or cardinality) can be proposed. So, the related examinations will not be launched.

- `TOOL_NAME`, that tells what tool is being processed by the system. This allows you to submit several tools (or variants of the same tool) hosted in the same image disk (then you have to specify this at submission time).

2.4 Answering to *B*ench*K*it and the Dedicated MCC'2013 Post-Analysis Scripts

Your tool must answer in STDOUT and may provide alternative messages on STDERR too. Both will be reported separately (this is useful for the debug phase).

However, there should be a dedicated line strictly respecting the format dedicated to a given examination. This line must start with the appropriate keyword (see below) that must no be show neither in STDOUT nor STDERR.

The format your tool must respect when answering the examinations planned in the MCC'2013 is presented below.

When an examination is not supported. The output on STDOUT must contain the following line:

`DO_NOT_COMPETE`

When the tool crashes. the output on STDOUT when you detect that your tool crashes must contain the following line:

`CANNOT_COMPUTE`

State Space generation. The output on STDOUT for this examination (in `benchKit_head.sh`, `BK_EXAMINATION="StateSpace"`) must contain the following line:

`STATE_SPACE` ⟨*number*⟩ `TECHNIQUES` ⟨$technique_1$⟩ ... ⟨$technique_n$⟩

where ⟨*number*⟩ is the number of states found in the Kripke structure.

and where ⟨$technique_i$⟩ denotes a value possibly picked in the set of values presented in table 1, page 5. Each technique must be separated by a space. You must specify several techniques if needed (up to 4 please).

Any Examination Involving Properties. The output on STDOUT for this examination (in `benchKit_head.sh`, `BK_EXAMINATION`≠"StateSpace") must contain the following line:

`FORMULA` ⟨*name*⟩ ⟨*res*⟩ `TECHNIQUES` ⟨$technique_1$⟩ ... ⟨$technique_n$⟩

where ⟨*name*⟩ is the name of the formula and ⟨*res*⟩ the result (`TRUE` or `FALSE`) of its evaluation.

There must be one such line per formula in the file. A very fine classification of formula is proposed so that tools can only participate when they are able to handle the corresponding atomic propositions. The idea is to answer such an examination if and only if all formula can be processed. Otherwise, just answer `DO_NOT_COMPETE` or `CANNOT_COMPUTE`.

Identifiers for Involved Techniques. Table 1 presents the list of identified techniques that could characterize your tool. If your tool uses a technique that is clearly not presented here, please add an appropriate keyword (one identifier, possibly containing the _ character) and provide us with a short explanation of this technique to update the table.

IMPORTANT: if your tool relies on another formalism than Petri net, you may provide the name of the formalism as a technique. Then, please provide it in the first position.

Value	Signification
ABSTRACTIONS	your tool exploits the use of abstractions (on the fly state elimination)
DECISION_DIAGRAMS	your tool uses any kind of decision diagrams
EXPLICIT	your tool does explicit model checking
NET_UNFOLDING	your tool uses MacMillan unfolding
PARALLEL_PROCESSING	your tool uses multithreading
STRUCTURAL_REDUCTION	your tool uses structural reductions (Berthelot, Haddad, etc.)
SAT_SMT	your tool uses a constraint solver
STATE_COMPRESSION	your tool uses some compression technique (other than decision diagrams)
STUBBORN_SETS	your tool uses partial order technique
SYMMETRIES	your tool exploits symmetries of the system
TOPOLOGICAL	your tool uses structural informations on the Petri net itself (e.g. siphons, traps, S-invariants or T-invariants, etc.) to optimize model checking
UNFOLDING_TO_PT	your tool transforms colored nets into their equivalent P/T

Table 1: List of possible techniques identified to characterize your tool

3 Creating your Own Disk Image

If you provide your own customized disk image, it must respect the following requirements:

- The logins *mcc* and *root* must be installed in the exact way they are in the disk image we distribute. In particular, the public ssh-key must be appropriately installed for the two logins and the machine must be reachable using `ssh`.

  If your disk image runs under Windows, you must install **Cygwin**[6] in order to let the invocation script be able to operate your tool. You must also integrate your tool to be launchable from the Unix mode proposed by **Cygwin**. An `ssh` server exactly as in the Linux configuration mush be installed too (to let us access the machine during your tool evaluation).

  Finally, the `NSCClient++` utility must be installed (contact us to get the parameters).

- You must untar, in the home directory, the content of the archive provided here: `http://mcc.lip6.fr/archives/MCC2013-TREE.bz2`.

- Install all packages required for your tool to run.

When creating your disk image, please be sure to let 5GBytes free in the filesystem. It is also advised that you avoid this image to be more than 2GBytes.

[6] `http://www.cygwin.com`