1 Introduction

This document presents the submission procedure of the Model Checking Contest @ Petri Nets 2012. Prior to any submission, please check that you are in the conditions of the Model Checking Contest @ Petri Nets 2012. 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.

2 What is provided in the Submission Kit

The submission kit is composed of the following elements:

- A public ssh-key to enable access to our invocation scripts,
- The MCC'2012 archive with model data and the invocation scripts,
- A disk image to be monitored in a virtual machine that contains the public key.

The public ssh-key. This is useful if you provide your own disk image that must strictly follow the requirements stated in this document. These requirements ensure communications between the MCC'2012 system invocation and your tool.

The MCC'2012 archive. It contains all models and examinations proposed in this contest[1]. The structure of this archive is detailed in section 4.1.

Important: at its publication time, the submission kit only contains the wrapper to your tool and the description of models. This is sufficient to start interfacing your tool with our invocation system (or upgrade the interface if you participated in the MCC'2011). Soon, an upgrade will be proposed including formulas to be evaluated. Please drop an email to `Fabrice.Kordon@lip6.fr` if you want to be informed as soon as the upgrade is out.

The disk Image. The one you can download from the MCC'2012 web site operates Linux Ubuntu server 11.04 with a full development environment. This is a standard distribution including a classic development environment. There are two logins: *mcc* and *root* that can be accessed using the public key during the evaluation of your tool. In both cases, the password is "mcc" for you.

This disk image will be operated as a QEMU/KVM virtual machine, the parameters to start the machine are provided also within the submission kit

You can either use this image to install your software following the MCC'2012 instructions below or elaborate your own (see section 3).

The dis-image does not contain the submission kit archive because it will be updated with properties files. You must uncompress this archive in the home directory of the `mcc` user.

3 Creating your Own Disk Image

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

[1] Except for the "push-button" category for which the model will be provided later, randomly selected from a set of existing models

- 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**[2] 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 copy, in the home directory of the *mcc* user, the MCC-2012 directory provided in the submission kit.

- 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 32GBytes.

4 Integrating your Tool in the Submission Kit

This section details the submission procedure and, in particular:

- the content of the submission kit archive,
- an overview of the execution procedure,
- the way to connect your tool to the execution script.

4.1 Content of the Archive

The home directory of the *mcc* user must contain the directory stored in the MCC'2012 archive (see figure 1) named MCC-2012 (see section 4.3 for possible customization of this directory). The invocation script will enter this directory to launch your model checker for all the examinations/models you have selected.

This directory contains:

- `benchmark.sh`: this file is the root script of the invocation system. It will be actually called from outside of the virtual machine and is provided for your information. You are not supposed to change it because it is very likely to be adapted when we will start to proceed the submissions. However, the virtual machine invocation itself is intended not to change in an essential way.

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

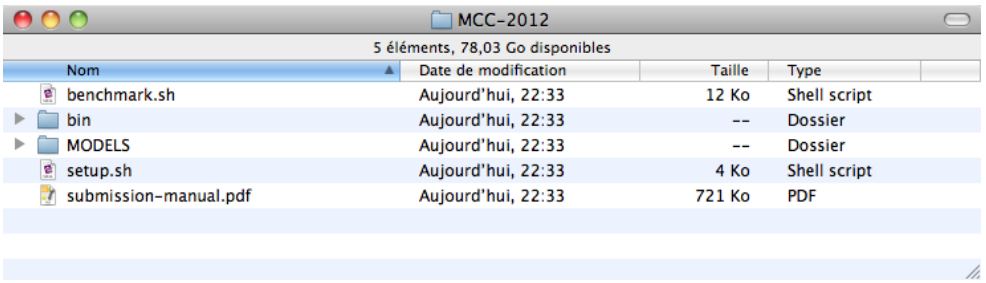

Figure 1: Structure of the MCC-2012 directory as it should be in the Home directory.

- `bin`: this directory is for the tool submitters. It will contain all the executable files required to operate your tool. All files in that directory should be in `r` (or `rx` when necessary) mode for the *mcc* user[3]. So far, this directory contains a dummy executable for the tool "`MY_TOOL`".
- `MODELS`: this directory contains models and their associated data for the MCC'2012. Models will be provided in the format they were submitted but also in PNML. Colored models will be provided in PNML for colored as well as PNML for their P/T net equivalent. Please note there may be missing instances for P/T net equivalents ; this is the case when the resulting P/T net is too large.
- `setup.sh`: this file is the interface between `benchmark.sh` and your tool. It is launched by the root through `ssh` script to operate your tool (the scripts will do).
- `submission-manual.pdf`: this document.

Be sure that user *mcc* is able to write in `$HOME/MCC-2012` and `$HOME/MCC-2012/MODELS`

4.2 Overview of the Execution Procedure

The MCC'2012 execution procedure mainly relies on two scripts: `benchmark.sh` and `setup.sh`. These scripts use the files located in `$HOME/MCC-2012/MODELS` (models description, formula files, etc.).

`benchmark.sh`. it is executed outside of the virtual machine. The version located in the archive may not be the final one since some details for the execution in the final cluster may change. We provide it for your information... and also to let you test how your tool will be operated in the final environment.

`setup.sh`. it is executed remotely on the virtual machine by `benchmark.sh`. This is the interface between our monitoring and launching mechanism and your tool. The version located in the archive is a "default" one that you must customize to interface your tool with our invocation system.

Tools Execution. For each examination (state space generation, evaluation of properties, etc.), and each model instance, `benchmark.sh`:

- 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[4] after retrieving the observed data and add them into a CSV file.

Please note that all tools will be operated in the same conditions to avoid any deviation in the measurement of CPU and memory.

4.3 Connecting Your Tool to the Execution Script

Important: The `benchmark.sh` and `setup.sh` scripts require the data located in `$HOME/MCC-2012/MODELS`. When submitting, you may add data that are suitable for your tool (typically, the models at the format it requires if PNML is not supported). However, please neither change the structure of this directory hierarchy, nor suppress any directory.

As mentioned in the previous section, `setup.sh` is the interface between our invocation mechanism and your tool. You must implement all the possible invocations of your tool and provide us with a standard answer (including the answer "I do not handle this examination on this model instance").

[3] `rwx` is also allowed ;-).

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

Communicating with Your Tool. `benchmark.sh` communicates with `setup.sh` by means of the following environment variables:

- `BASE_DIR`: it is the base directory of your submission,
- `BIN_DIR`: it is the directory containing the binaries of your tool,
- `MODELS_DIR`: it is the directory containing the directories for all models,
- `MODEL_DIR`: its value is either `COLORED` or `PT` according to the type of model being processed,
- `MODEL`: it is the name of the model currently being processed,
- `MODEL_TYPE`: it is the type of the model currently being processed. Possible values for this variable are:

Value	Signification
`COLORED_EXTRA`	It is a colored model, but with extra features that may not be supported by any tool ; equivalent P/T unfolded model is provided too
`COLORED`	It is a colored model ; equivalent P/T unfolded model is provided too
`PT`	It is a P/T Model

  Note that, in the case of Colored Petri nets some instances of the colored model may not exist in P/T. This is the case when the P/T equivalent net is too large. This situation is tagged with a dedicated environment variable `EQUIV_PT`.

- `EQUIV_PT`: when its value is `YES`, the P/T net equivalent to the Colored Net being processed does exist. Otherwise, due to its size, the value is `NO`. Then, your tool just have to report this situation in the standard way (see "Possible Invocations of your Tool").
- `SCALE`: it is the current scaling value for the model currently being processed,
- `FORMULA_FILE`: when evaluating formulas (option `-check`), it contains the path of formula file currently being processed,
- `FORMULA_TYPE`: when evaluating formulas (option `-check`), it contains the type of formula to be processed to let your tool decide whether it can support these or not. Possible values for this variable are:

Value	Signification
`REACHABILITY`	It means that the file contains reachability formulas only
`LTL`	It means that the file contains LTL formulas only
`CTL`	It means that the file contains CTL formulas only
`STRUCTURAL`	It means that the file contains structural formulas only

  We call:

  - *reachability formulas* the ones that only contain a combination of logic propositions on place marking,
  - *LTL formulas* the ones that only relate logic propositions on place marking with LTL operators,
  - *CTL formulas* the ones that only relate logic propositions on place marking with LTL operators,
  - *structural formulas* the ones that refer to structural properties such as the presence of deadlocks, bounds of places, etc.

- `NEW_MODEL`: models provided to you are presented in the submission kit, except for a special category (containing just one model) that is not known in advance and aims at evaluating the capability of your tool with its "defaults parameters". This model is provided in PNML only and, in this case, this environment variable is set to `TRUE` (`FALSE` otherwise).

Possibles Invocations of your Tool. `setup.sh` usually provides your tool answer via `stdout`. Two default values must be provided in the following situations:

- when your tool ***does not compete*** for an examination or a model or a model instance, `setup.sh` must return via `stdout` the following string:

  `DO_NOT_COMPETE`

- when you detect that your tool ***cannot process the query*** for an examination of a model instance, `setup.sh` must return via `stdout` the following string:

  `CANNOT_COMPUTE`

- when you work on the P/T equivalent net of a colored one and the corresponding description is not provided (this is indicated thanks to an environment variable), then `setup.sh` must return via `stdout` the following string:

  `NO_EQUIV_PT`

`setup.sh` can be invoked using the following parameters:

- `-info`: this parameter allows our invocation mechanism to extract data concerning your tool (submitter's name, institution, URL, etc.). The information must be provided back by means of environment variables.

  `setup.sh` must always answer to this question.

- `-generate`: this option requires your tool to generate the state space. If your tool supports this examination, you must provide the answer to `benchmark.sh` via `stdout` using the following format:

  `STATE_SPACE` <*Colored*> <*SVAL*> `TECHNIQUES` <$TVAL_1$> <$TVAL_2$> ...

  <*Colored*> is set to `TRUE` if you computed the state space on the Colored Net[5] or false otherwise. <*SVAL*> is the number of states your tool have computed. For the value of <$TVAL_i$>, see section 4.4.

- `-check`: this option requires your tool to process a file containing several formulas (using the MCC'2012 dedicated syntax). If your tool supports this examination, you must provide the answer to `benchmark.sh` via `stdout`. For each formula in the file, there is a line that must have the following format:

  `FORMULA` <*Colored*> <*NAME*> <*RES*> `TECHNIQUES` <$TVAL_1$> <$TVAL_2$> ...

  <*Colored*> is set to `TRUE` if you computed the state space on the Colored Net[6] or false otherwise. <*NAME*> is the name of the evaluated formula, and <*RES*> is the value `TRUE` if there the formula is verified (no counter-example) by the system, `FALSE` otherwise. For the value of <$TVAL_i$>, see section 4.4.

Structure of the `MODELS` directory. The `MODELS` directory contains one subdirectory per model (called a *"model directory"*). A model directory contains the following data:

- the model form and its associated sources (the model form is also available in the MCC'2012 web site),
- the `PNML` directory that itself contains two directories: `COLORED` for colored PNML files of colored models[7] and `PT` that for either the PNML of P/T models, or the PNML of P/T equivalent models for colored models.
- the `PROPRIETARY` directory that contains proprietary formats. There is no predefined structure for this directory. Please insert your files there if your tool does not support PNML.

[5] If the current model is a P/T net, then, you always return `FALSE`.
[6] If the current model is a P/T net, then, you always return `FALSE`.
[7] This directory exists only when the model is colored.

- the `PROPERTIES` directory that contains properties to be in evaluated several files.

There is a convention that enables you to build the path of the file describing a model by using the environment variables in the following way:

- for models with no scaling parameter:

  `$MODELS_DIR/PNML/<`*TYPE*`>/$MODEL.pnml`

- for models with a scaling parameter:

  `$MODELS_DIR/PNML/<`*TYPE*`>/$MODEL-$SCALE.pnml`

<*TYPE*> is either `COLORED_EXTRA`, `COLORED` or `PT` according to the type of model your tool uses for computation (Colored, P/T, equivalent P/T, etc.).

Similar conventions may be used for proprietary formats, but inside the `PROPRIETARY` folder.

About the "Push Button" mode. to evaluate the capabilities of tools in a "push-button" mode, some case studies models are proposed. They will be displayed after the submission ends. For the first year of this category, we will propose P/T models only in PNML format (the easiest to analyze). Tools are welcome to participate in this category, and thus, to parse P/T PNML files.

4.4 List of Techniques to be identified

It is important to mention the techniques used by a tool to solve an examination. This allows a better identification for analysis purpose. To normalize the values and avoid the multiplicity of terms, here is the list of constants that are allowed in the <$TVAL_i$> elements of your answer in `setup.sh`:

- **ABSTRACTIONS**: if your tool exploits the use of abstractions (on the fly state elimination),
- **DECISION_DIAGRAMS**: if your tool uses any kind of decision diagrams,
- **EXPLICIT**: for explicit model checking,
- **NET_UNFOLDING**: if your tool uses MacMillan unfolding,
- **PARALLEL_PROCESSING**: if your tool uses multithreading,
- **STRUCTURAL_REDUCTION**: if your tool uses structural reductions (Berthelot, Haddad, etc.),
- **SAT_SMT**: if your tool uses a constraint solver,
- **STATE_COMPRESSION**: if your tool uses some compression technique
- **STUBBORN_SETS**: if your tool uses partial order technique
- **SYMMETRIES**: if your tool exploits symmetries of the system
- **TOPOLOGICAL**: if 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**: if your tool transforms colored nets into their equivalent P/T,
- **OTHERS**: for all unlisted technique ; please then characterized it by a single word identifier (use of "_" allowed)).