Model Checking Contest 2020

Tool Submission Manual

http://mcc.lip6.fr

Contents

1 Introduction

This document presents the tool submission procedure of the Model Checking Contest @ Petri Nets 2020. Prior to any submission, please check that you meet the conditions of the Model Checking Contest @ Petri Nets 2020. These rules are available at `http://mcc.lip6.fr/rules.php`. These rules now also contain information about the way we execute and evaluate tools during the Model Checking Contest.

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

IMPORTANT: this year, sequential tools will be processed on virtual machines having 1 core and 16GByte of memory and parallel tools will be executed on virtual machines with 4 cores and 16 GByte of memory. In both cases, the planned confinement is 60 minutes per run (one examination per instance of a model).

IMPORTANT: this year, even if all the "known models" are provided in the submission kit, only a subset of these (at least one instance per model) will be executed during the contest. This is to keep an appropriate balance between new models and existing ones.

About the Execution Environment.

To improve the tool integration procedure, we developed a simple and separate benchmarking environment to:

- enable one to reproduce the experience since, if you agree, the submitted VM will be made publicly available,
- ease the work of tool developers when building their tool submission.

BenchKit[1], is the benchmarking environment that will be used for tool evaluation during the Model Checking Contest. Introduced in the 2013 edition, it has been enhanced for the 2014, 2015, 2016, 2017, 2018 and 2019 editions. The tool submission kit embed simplified scripts from *BenchKit*. If **KVM/Qemu**[2] or **VirtualBox**[3] is installed, then you may operate and thus test the VM of your tool(s) in similar conditions than the ones of the MCC.

So, your tool submission must be a disk image to be executed in a virtual machine. The disk image preferred format is `.vmdk` that is compatible at least with **KVM/Qemu** and **VirtualBox**.

IMPORTANT: Please name your VM in the following way: ⟨*toolName*⟩-⟨*year*⟩.vmdk. This will help us to avoid mistakes in manipulating the virtual machines.

This year, we provide you with two ways to experiment the integration of your tool with *BenchKit*. First, the use of the "personal version" as for the past years (see section 3.1), and second, the possibility to execute your tool in similar conditions than those of the MCC with the "full version" of *BenchKit* (see section 3.2, this is of interest if you have access to a powerful multicore machine). You can then extract information to be processed for checking purposes and also get monitoring data such as CPU or memory consumption.

2 Description of the 2020 Tool Submission Kit

This section presents the structure of the Tool Submission Kit and ends with a procedure to let you integrate your tool for a proper invocation during the evaluation phase of the MCC'2020.

2.1 Overview

Tools are operated in Virtual Machines (VM). Thus, the tool submission must be a disk image containing your tool. By default, Linux is planned, and thus, the disk image we provide operates a Linux machine, if you need another distribution or another operating system, please have a look at section 4, page 10.

[1] *BenchKit* (`http://BenchKit.CosyVerif.org`) is developed within the context of the **CosyVerif** project (`http://CosyVerif.org`), supported by the **MeFoSyLoMa** group (`http://www.mefosyloma.fr`).

[2] `http://wiki.qemu.org`.

[3] `https://www.virtualbox.org`.

The tool submission kit is composed of the following elements:

- `mcc2020.vmdk` and `mcc2020-input.vmdk`, two disk images. `mcc2020.vmdk` is a booting image, preinstalled with a dummy tool allowing tool developers to become familiar with how the system works. This dummy tool only supports the State Space examination for the first instance of each model and returns a result validated by tools in the previous years.

  `mcc2020-input.vmdk` contains models and formulas only, it is mounted by the VM implemented in `mcc2020.vmdk` at `/Home/mcc/BenchKit/INPUTS` in readonly mode.

- the private `ssh` key (file `bk-private_key`) associated with the two accounts installed in the virtual machine (`mcc` and `root` are configured to log in with this key[4] – an empty passphrase is associated to this key). To connect with a password only, the password is: `mcc,2020`. Never remove this key because it will be used to operate your tool during the evaluation phase.

- a distribution of the "full version" of *BenchKit* to be operated with **KVM/Qemu** on a Linux machine. Section 3.2, page 8, shows how to use this environment to test the behavior of the VM automatically in the conditions of the MCC'2020. You may alternatively use a previous distribution of *BenchKit* that ease the test of your tool in the environment of the MCC but is less efficient (see section 3.1, page 7).

- the formula manual that presents the structure of the XML files to be parsed when your tool evaluates formulas.

BenchKit and the provided disk images are ready to be used together, provided that you adapt the description of the disk image to be used.

2.2 Content of the boot Disk Image

The `mcc2020.vmdk` disk image embeds a Linux system (Debian 10 – Buster stable – in 64 bit mode) including two accounts: `root` and `mcc`. The `mcc` home directory is structured as presented in figure 1. You find a unique directory, `BenchKit`, containing:

- `bin`, a directory where you should put all the executables and libraries of your tool. A "dummy tool" is provided as an example. It only supports state space generation for the first instances of each "known" model,

  IMPORTANT: You are free to install whatever you want in this directory that must contain all the libraries, executable files and data required to operate your tool.

- `INPUTS`, a directory stored in a separate disk image, `mcc2020-input.vmdk`, that contains all the instances of models (there is one instance of model per value of the scaling parameters, only one when the model has no scaling parameter) 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 2.5.

  IMPORTANT: Benchmark models are provided in a compressed archive to reduce the size of the disk image (one per model and per instance). Each archive contains all the required data for a given test (PNML files, formulas, etc.) When evaluating your tool for a given model instance and a given examination, *BenchKit* will uncompress the corresponding archive in a temporary directory and execute your tool in that directory. See section 2.5, page 4 for more details on the content of each compressed directory.

- `BenchKit_head.sh`, a script executed remotely in the virtual machine; it is dedicated to the invocation of your tool.

  In `BenchKit_head.sh`, you may access to the following environment variables providing useful information:

[4] To do so, start your connection as follows: `ssh -i bk-private_key ...`

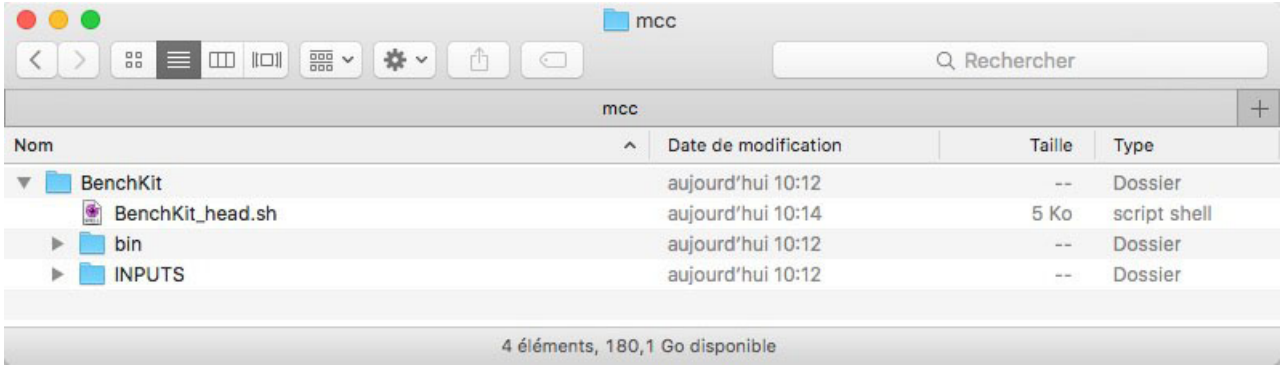


Figure 1: Structure of the `mcc` user home directory in the disk image.

- `BK_TOOL` that names the invoked tool (useful when you provide several tool variants in the same virtual machine),
- `BK_TIME_CONFINEMENT` that shows the time confinement in seconds,
- `BK_MEMORY_CONFINEMENT` that shows the memory confinement in Mbytes.

IMPORTANT: You must adapt this script that will be used by *BenchKit* to invoke appropriately your tool since the evaluation environment will only knows it (and not command-line needed to run your tool). Two environment variables (see section 2.5) help you to determine which examination is being processed and, if several tools are hosted in the same VM, the tool to be invoked.

2.3 Overview of the Execution Procedure

The MCC'2020 execution procedure relies on *BenchKit*. 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, *BenchKit*:

- starts the virtual machine and uncompresses the data required to operate the current examination,
- operates CPU and memory monitoring in the virtual machine,
- operates your tool for the examination on the model instance,
- stops the virtual machine[5] after retrieving the observed data and records 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.

2.4 Connecting and Upgrading the Virtual Machine From the Disk Image

This section explains how you may operate the virtual machine from the `mcc2020.vmdk` disk image we provide in the submission kit. It is important to follow the integration directives provided in section 2.5.

To install extra software, you have access to the `root` account in this VM (it has the same public key and password than the `mcc` account – see page 4). You must also install your stuff (binaries, extra model descriptions, etc.) in the `mcc` home directory (see section 2.5).

Starting the Virtual Machine. The following command starts a VM with your copy of the disk image (let's call it `my-disk-image.vmdk`) and a reference to the `mcc2020-input.vmdk` disk image containing the benchmark for the MCC , please type:

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

```
$ qemu-system-x86_64 -vnc :42 -enable-kvm -hdb mcc2020-input.vmdk -smp 1 -cpu Westmere \
  -daemonize -k fr -m 2048 -net nic -net user,hostfwd=tcp::2222-:22 my-disk-image.vmdk
```

You may want to omit the `-demonize` option to get the default screen where outputs of your VM are propagated. You may also connect using a VNC server (port 42).

This command supposes that you have **KVM/Qemu** but a similar operation can be done with **VirtualBox** (you must however set-up port redirection by means of the user interface before starting the VM).

Of course, you will adapt the `-k` parameter to your keyboard.

If you want to connect without using ssh keys, please note that the **mcc** password is : `mcc,2020`.

Connecting to the Virtual Machine. Once the VM is running, you must type to log in:

```
$ ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -p 2222 -i bk-private_key mcc@localhost
```

where `bk-private_key` contains the private key associated to the public key installed for `mcc` (it is provided in the tool submission kit). The same couple of keys stand for `root`, thus allowing you to install packages if needed.

Please note that the options "`-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no`" can be ignored. It is useful to avoid updating your `known_hosts` file and to have to change it if you manage several virtual machines with the same redirection port.

Copying files to/from the Virtual Machine. Once the VM is running, you must type to copy files:

```
$ scp -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -P 2222 <files> mcc@localhost:<location>
```

where `<files>` represent your files and `<location>` the target destination of these files in the Virtual machine.

If you want to copy files without using ssh keys, please note that the **mcc** password is : `mcc,2020`.

2.5 Connecting Your Tool to the Execution Script

Since the connection of your tool to the execution system is based on the same principle as for the past editions, its adaptation for people who already participated in previous MCC should not be a problem. *BenchKit*, the execution environment sets-up environment variables to let you know what to do and launches your tool in a directory with all the appropriate data in.

Environment variables. There are two environment variables set-up for you by *BenchKit*:

- `BK_EXAMINATION`, that specifies the examination we expect your tool to perform. Possible values for this variable are presented in table 1, page 5. Please have a look on the property language manual for more details on the corresponding classification.
- `BK_TOOL`, 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 clearly specify this when submitting your tools).

Content of the directory where your tool is being executed. As mentioned before, `BenchKit_head.sh` is operated by *BenchKit* to run your tool in a directory that contains all you need to compute the current examination for one instance of a model:

- `model.pnml`, is the initial PNML file that you may use for the model checking contest,
- `iscolored`, contains either `TRUE` (if this is a colored model) or `FALSE` (if it is a P/T one),

- ⟨category⟩.xml and ⟨category⟩.txt where ⟨category⟩ is the value of the BK_EXAMINATION environment variable when it designates an examination where properties are required.

  Please note that the textual version of the formula is mainly to ease reading. As explained in the formula manual, the XML grammar of formula files only is provided (it is simpler to maintain and to use).

- 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 or P/T net),
- instance, a file containing the value of the current instance for the model,
- unfinite, a empty file present when the model shows unfinite marking graph if we are aware of this[6],
- large_marking, a empty file present when the model has at least one marking containing more than 2^{32} tokens if we are aware of this[7],
- GenericPropertiesDefinition.xml, a file containing the list of precomputed properties (from the model form),
- GenericPropertiesVerdict.xml, a file containing the value of each precomputed properties. For example, it is easy to check if the current model is safe or not by just using the following bash command:

  ```
  grep SAFE GenericPropertiesVerdict.xml | cut -d \" -f 6
  ```

  You will find all the informations about these precomputed properties in the rules[8] (section I, rule M-9, page 2) or in http://mcc.lip6.fr/verdict-properties.php.

For "known" Models . You are not allowed to enrich the content of this directory. **Remind that precomputation of results is not allowed, only the information stored in model forms can be exploited.**

For "surprise" Models. "Surprise" models are meant to evaluate tools in a default mode (i.e. no dedicated optimization) and on models that discovered after the tool evaluation sarts. Thus, they are not present in the initial archive provided to you but they have the exact same structure and an empty file named NewModel will be located in the directory in that case to let your tool detect such situations. In

[6] for the MCC'2020, this will not be present in the initial disk image provided with the submission kit but will be generated in the one produced for the execution for the contest.

[7] for the MCC'2020, this will not be present in the initial disk image provided with the submission kit but will be generated in the one produced for the execution for the contest.

[8] See http://mcc.lip6.fr/pdf/rules.pdf

Value	Signification
StateSpace	We ask for state space generation only
UpperBounds	Wa ask to computing the exact upper bounds of places
ReachabilityDeadlock	We ask for a global properties of the model : deadlock detection
QuasiLiveness	We ask for a global properties of the model : quasi-liveness detection
StableMarking	We ask for a global properties of the model : stable marking detection
Liveness	We ask for a global properties of the model : liveness detection
OneSafe	We ask for a global properties of the model : 1-safe detection,
ReachabilityFireability	Boolean combinations of propositions checking firability of transitions
ReachabilityCardinality	Boolean combinations of propositions comparing the number of tokens in places
LTLFireability	Full LTL with atomic propositions checking firability of transitions
LTLCardinality	Full LTL with atomic propositions comparing the number of tokens in places
CTLFireability	Full CTL with atomic propositions checking firability of transitions
CTLCardinality	Full CTL with atomic propositions comparing the number of tokens in places

Table 1: Possible values to refer to examinations in the environment variable BK_EXAMINATION

fact, the MCC organizers will use a dedicated version of the `mcc2020-input.vmdk` file that will contain all models and their formulas computed for 2020.

IMPORTANT: 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.

2.6 Answering to *BenchKit* and the Dedicated MCC'2020 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 dedicated keywords (see below). These keywords must never appear on the head of a line, neither in `stdout` nor `stderr`.

IMPORTANT: in any case, please always mention the type of processing (see the three values proposed in the top of the table 2, page 8). If several results are provided you may have a different specification for each provided value. This helps a deeper analysis of the outputs for the whole community.

The format your tool must respect when answering the examinations 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 (and you detect it). 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 lines:

`STATE_SPACE STATES` $\langle num_S \rangle$ `TECHNIQUES` $\langle technique_1 \rangle$ ... $\langle technique_n \rangle$

`STATE_SPACE TRANSITIONS` $\langle num_T \rangle$ `TECHNIQUES` $\langle technique_1 \rangle$ ... $\langle technique_n \rangle$

`STATE_SPACE MAX_TOKEN_PER_MARKING` $\langle num_M \rangle$ `TECHNIQUES` $\langle technique_1 \rangle$ ... $\langle technique_n \rangle$

`STATE_SPACE MAX_TOKEN_IN_PLACE` $\langle num_P \rangle$ `TECHNIQUES` $\langle technique_1 \rangle$ ... $\langle technique_n \rangle$

where $\langle num_S \rangle$ is the number of states found in the marking graph, $\langle num_T \rangle$ is the number of transitions firing (with duplicates, see the MCC'2020 rules) in the marking graph, $\langle num_M \rangle$ is the maximum number of tokens per marking in the net, $\langle num_P \rangle$ is the maximum number of token that can be found in a place and where $\langle technique_i \rangle$ describes the verification technique(s) that has(ve) been used by your tool. Please pick one value in the set presented in table 2, page 8. Techniques must be separated by a space. You may specify several techniques if needed (no more than 8 please).

IMPORTANT: These lines can be provided in any order.

IMPORTANT: However, only the computation of $\langle num_S \rangle$ is mandatory. Tools may answer -1 for $\langle num_T \rangle$, $\langle num_M \rangle$, and $\langle num_P \rangle$ if they cannot compute the information (providing these values brings a bonus).

UpperBounds. The output on `stdout` for this examination (in `benchKit_head.sh`, `BK_EXAMINATION="UpperBounds"`) must contain the following lines:

`FORMULA` $\langle name \rangle$ $\langle int \rangle$ `TECHNIQUES` $\langle technique_1 \rangle$ ... $\langle technique_n \rangle$

where $\langle name \rangle$ is the formula identifier (provided in both the XML and textual format, see the formula manual for more details) and $\langle int \rangle$ the result of the formula: an integer value representing the upper bound of the designated place.

There must be one such line per formula. A classification of formulas is proposed so that tools should only participate when they can handle the class of formulas that will be presented in this file (see the property language manual for more details about the subcategories of the contest).

For each formula your tool has a problem with, please return:

FORMULA ⟨*name*⟩ CANNOT_COMPUTE

IMPORTANT: never answer DO_NOT_COMPETE for a given formula since this keyword must be only used to state that a tool does not participate in the whole subcategory.

Reachability, CTL and LTL Examinations. The output on `stdout` for these examination (in `bench- Kit_head.sh`, BK_EXAMINATION≠"StateSpace") must contain the following line:

FORMULA ⟨*name*⟩ ⟨*bool*⟩ TECHNIQUES ⟨$technique_1$⟩ ... ⟨$technique_n$⟩

where ⟨*name*⟩ is the formula identifier (provided in both the XML and textual format, see the formula manual for more details), and ⟨*bool*⟩ the result of the formula: TRUE or FALSE.

There must be one such line per formula. A fine classification of formulas is proposed so that tools should only participate when they can handle the class of formulas that will be presented in his file (see the property language manual for more details about the subcategories of the contest).

For each formula your tool has a problem with, please return:

FORMULA ⟨*name*⟩ CANNOT_COMPUTE

IMPORTANT: never answer DO_NOT_COMPETE for a given formula since this keyword must be only used to state that a tool does not participate in the whole subcategory.

Identifiers for Involved Techniques. Table 2 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 put it in the first position.

IMPORTANT: For examinations with several formulas, please select the exact techniques used to solve each formula and not all the potential techniques your tool operates.

GlobalProperties (ReachabilityDeadlock, QuasiLiveness, StableMarking, Liveness, OneSafe). The output on `stdout` for these examination follows the structure of formulas but withe the name of the examination used as the formula identifier since there is only one answer expected. So, it must contain the following line:

FORMULA ⟨*examinationName*⟩ ⟨*bool*⟩ TECHNIQUES ⟨$technique_1$⟩ ... ⟨$technique_n$⟩

where ⟨*examinationName*⟩ is the name of the examination (stored in the environment variable BK_EXAMINATION) used as the formula identifier, and ⟨*bool*⟩ the result of the formula: TRUE or FALSE.

There must be one such line per formula. A fine classification of formulas is proposed so that tools should only participate when they can handle the class of formulas that will be presented in his file (see the property language manual for more details about the subcategories of the contest).

For each formula your tool has a problem with, please return:

FORMULA ⟨*examinationName*⟩ CANNOT_COMPUTE

Remind that the examination name is provided in the environment variable BK_EXAMINATION.

3 Testing the Virtual Machine in the MCC'2020 Conditions

To launch an execution of your tool in the conditions of the MCC'2020, you must let the structure of the tool submission kit unchanged and be at the root of the uncompressed archive.

3.1 The "personal" version of *BenchKit*

The main script to be used is `BenchKitStart.sh`. This script boots a VM with your disk image, then operate your tool for a given examination on a given model and then stops the VM and display the outputs

Value	Signification
SEQUENTIAL_PROCESSING	your tool is sequential
PARALLEL_PROCESSING	your tool is intrinsically parallel (it implements parallel algorithms)
COLLATERAL_PROCESSING	your tool is parallel but with a portfolio-like approach
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
IMPLICIT	your tool handles implicit representations of states other than Decision Diagrams
NET_UNFOLDING	your tool uses McMillan unfolding
UNFOLDING_TO_PT	your tool transforms colored nets into their equivalent P/T
QUERY_REDUCTION	any pre-analysis technique allowing simplification of formulas (tautology, check on the initial state, etc.)
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
USE_NUPN	the PNML model contains a NUPN-toolspecific section (see `http://mcc.lip6.fr/nupn.php`) and your tool takes advantage of it

Table 2: List of possible techniques identified to characterize your tool. If some technique you use is not referenced, please contact us.

of your tool.

To be operated, it requires either **KVM/Qemu** or **VirtualBox** to be installed on your machine. Then, you can check the behavior of your tool in the conditions of the MCC'2020. You can also check that outputs conform to the expectations of section 2.6.

It requires four mandatory parameters:

- the path of the disk image to be booted and executed by the VM containing your tool,
- the value to assign to the `BK_EXAMINATION` environment variable defining what operation is to be executed on the VM (see table 1, page 5),
- the name of the tool to be invoked,
- the name of models to be processed with the examination (possible values are provided in table 3, Appendix A, page 11).

Thus, a typical invocation is:

`./BenchKitStart.sh` ⟨*vmname*⟩`.vmdk` ⟨*examination*⟩ ⟨*toolName*⟩ ⟨*modelName*⟩

A full execution example is provided in Appendix B, page 11.

3.2 The "full" version of *BenchKit*

You will find enclosed in the Submission Toolkit (directory `BenchKit2`) a variant of *BenchKit* that is (to some minor details) the one we use for the contest. All instructions to deploy and use it in your environment is detailed in the `README.txt` file.

To install *BenchKit*, type `make` and the install/uninstall documentation will be displayed.

```
$ make
================================================================================
 BenchKit version 2 (2020)
 F. Kordon and F. Hulin-Hubard
================================================================================

Possibles options are:

 - install     : generate links for an easy access to all BenchKit commands
 - long        : same as install but with long command names
 - uninstall   : uninstall BenchKit (delete links)
 - clean       : same as uninstall
```

The best configuration to operate *BenchKit* is to have two machines: one where the scripts will be executes (called *head machine*) and the one (possibly a multi-core machine with large memory) where the virtual machines will be executed (called *execution machine*). The two machines can be the same (this has never been extensively tested and might not be comfortable if you are using this only machine during the execution of our tool). These scripts work under Linux with **KVM/Qemu** (tested on version 1.5.3).

BenchKit is operated thanks to the `Makefile-runs` file. Please type:

- `make -f Makefile-runs` to get the list of possible entries in the makefile,
- `make -f Makefile-runs deploy_bk` to install BenchKit in the root directory of the execution machine,
- `make -f Makefile-runs deploy_vm` to deploy your disk imaqe in the target directory on the execution machine,
- `make -f Makefile-runs create_bench` to generate a sequence of executions for your tool (also called "slides of bench") described in a file with the `.bkjobs` extension,
- `make -f Makefile-runs execute_bench` to execute the sequence of executions described in the corresponding `.bkjobs` file in the execution machine.

You must first configure the `Makefile-runs` by setting some environment variables (all constants are located at the top of the file). You may set-up:

- the name of the benchmark (or slice) in variable `RUNNAME`,
- your email to be notified once the execution is terminated (required mail to be configured on the execution machine) in variable `EMAIL`,
- the directory, on the head machine, where all the disk images are located in variable `IMAGE_DIR`,
- the time confinement for the execution of the tool in variable `CONF_TIME`,
- the memory confinement for the execution of the tool in variable `CONF_MEM`,
- the number of core to be allocated to the virtual machine executing your tool in variable `CONF_CORES`,
- the address of the execution machine in variable `EXECMACHINE`,
- the login you use on the execution machine (acces via ssh must be set-up) in variable `EXECLOGIN`,
- the directory where all data are stored (disk images and execution outputs) on the execution machine in variable `EXECVMDIR`,

- the number of parallel executions of your tool you expect on the execution machine in variable `NBVNODES`.

Three files also allow to configure the slide of executions:

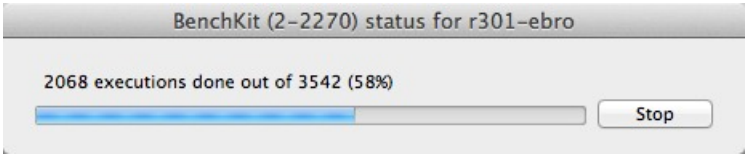
- `tools_all.txt` that lists the tools to be executed (if you have several)
- `models_selected.txt` that lists the models to be processed
- `examinations_all.txt` that lists the examinations to be processed

A default version is provided for the two last files that consider all "known" models and all examinations.

When a slice is being executed on the execution machine, you may follow its progress using the command `bk_monitor`. For example, the command:

```
./bkmonitor mybenchmark.bkjobs
```

provides the following information (in graphical mode if the head machine runs macOS).



An email with instructions to retrieve the collected data (monitored data from CPU and memory, execution logs) as soon as the execution of the slice ends.

More informations about *BenchKit* can be found in: F. Kordon and F. Hulin-Hubard. BenchKit, a Tool for Massive Concurrent Benchmarking. In 14th International Conference on Application of Concurrency to System Design, ACSD 2014, Tunis La Marsa, Tunisia, June 23-27, 2014, pages 159–165, 2014 (available at `http://lip6.fr/Fabrice.Kordon/pdf/2014-ACSD.pdf`).

4 Creating your Own Disk Image

When creating your disk image, please be sure it emulates a 64bits machine and has at least 5 GBytes free in the filesystem (for security matters if some models produce large temporary files when monitoring their execution). It is also advised that you avoid this image to be more than 2 GBytes.

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, please contact us (`Fabrice.Kordon@lip6.fr` and `Francis.Hulin-Hubard@llip6.fr`)
- You must untar, in the home directory, the content of the archive provided here: `http://mcc.lip6.fr/archives/MCC-INPUTS.tgz`. It contains the structure of the input models to be installed in the `mcc` account. You must add the `bin` directory as well as your copy of the `BenchKit_head.sh` file.
- Install all packages required for your tool to run.

A Appendix – the names of "known" models

This appendix displays all the 1022 model/instances that are provided this year in the "known" model category. It allows you to check automatically the behavior of your tool submission in the conditions of the Model Checking Contest. Please remind that only a subset of these will be executed during the contest.

Name of the model/instances ("known" models)	
ARMCacheCoherence-PT-none	ASLink-PT-01a
ASLink-PT-01b	ASLink-PT-02a
ASLink-PT-02b	ASLink-PT-03a
ASLink-PT-03b	ASLink-PT-04a
ASLink-PT-04b	ASLink-PT-05a
ASLink-PT-05b	ASLink-PT-06a
ASLink-PT-06b	ASLink-PT-07a
ASLink-PT-07b	ASLink-PT-08a
ASLink-PT-08b	ASLink-PT-09a
ASLink-PT-09b	ASLink-PT-10a
ASLink-PT-10b	AirplaneLD-COL-0010
AirplaneLD-COL-0020	AirplaneLD-COL-0050
AirplaneLD-COL-0100	AirplaneLD-COL-0200
AirplaneLD-COL-0500	AirplaneLD-COL-1000
AirplaneLD-COL-2000	AirplaneLD-COL-4000
AirplaneLD-PT-0010	AirplaneLD-PT-0020
AirplaneLD-PT-0050	AirplaneLD-PT-0100
AirplaneLD-PT-0200	AirplaneLD-PT-0500
AirplaneLD-PT-1000	AirplaneLD-PT-2000
AirplaneLD-PT-4000	Angiogenesis-PT-01
Angiogenesis-PT-05	Angiogenesis-PT-10
Angiogenesis-PT-15	Angiogenesis-PT-20
Angiogenesis-PT-25	Angiogenesis-PT-50
AutoFlight-PT-01a	AutoFlight-PT-01b
AutoFlight-PT-02a	AutoFlight-PT-02b
AutoFlight-PT-03a	AutoFlight-PT-03b
AutoFlight-PT-04a	AutoFlight-PT-04b
AutoFlight-PT-05a	AutoFlight-PT-05b
AutoFlight-PT-06a	AutoFlight-PT-06b
AutoFlight-PT-12a	AutoFlight-PT-12b
AutoFlight-PT-24a	AutoFlight-PT-24b
AutoFlight-PT-48a	AutoFlight-PT-48b
AutoFlight-PT-96a	AutoFlight-PT-96b
BART-COL-002	BART-COL-005
BART-COL-010	BART-COL-020
BART-COL-030	BART-COL-040
BART-COL-050	BART-COL-060
BART-PT-002	BART-PT-005
BART-PT-010	BART-PT-020
BART-PT-030	BART-PT-040
BART-PT-050	BART-PT-060
BridgeAndVehicles-COL-V04P05N02	BridgeAndVehicles-COL-V10P10N10
BridgeAndVehicles-COL-V20P10N10	BridgeAndVehicles-COL-V20P10N20
BridgeAndVehicles-COL-V20P10N50	BridgeAndVehicles-COL-V20P20N10

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
BridgeAndVehicles-COL-V20P20N20	BridgeAndVehicles-COL-V20P20N50
BridgeAndVehicles-COL-V50P20N10	BridgeAndVehicles-COL-V50P20N20
BridgeAndVehicles-COL-V50P20N50	BridgeAndVehicles-COL-V50P50N10
BridgeAndVehicles-COL-V50P50N20	BridgeAndVehicles-COL-V50P50N50
BridgeAndVehicles-COL-V80P20N10	BridgeAndVehicles-COL-V80P20N20
BridgeAndVehicles-COL-V80P20N50	BridgeAndVehicles-COL-V80P50N10
BridgeAndVehicles-COL-V80P50N20	BridgeAndVehicles-COL-V80P50N50
BridgeAndVehicles-PT-V04P05N02	BridgeAndVehicles-PT-V10P10N10
BridgeAndVehicles-PT-V20P10N10	BridgeAndVehicles-PT-V20P10N20
BridgeAndVehicles-PT-V20P10N50	BridgeAndVehicles-PT-V20P20N10
BridgeAndVehicles-PT-V20P20N20	BridgeAndVehicles-PT-V20P20N50
BridgeAndVehicles-PT-V50P20N10	BridgeAndVehicles-PT-V50P20N20
BridgeAndVehicles-PT-V50P20N50	BridgeAndVehicles-PT-V50P50N10
BridgeAndVehicles-PT-V50P50N20	BridgeAndVehicles-PT-V50P50N50
BridgeAndVehicles-PT-V80P20N10	BridgeAndVehicles-PT-V80P20N20
BridgeAndVehicles-PT-V80P20N50	BridgeAndVehicles-PT-V80P50N10
BridgeAndVehicles-PT-V80P50N20	BridgeAndVehicles-PT-V80P50N50
BusinessProcesses-PT-01	BusinessProcesses-PT-02
BusinessProcesses-PT-03	BusinessProcesses-PT-04
BusinessProcesses-PT-05	BusinessProcesses-PT-06
BusinessProcesses-PT-07	BusinessProcesses-PT-08
BusinessProcesses-PT-09	BusinessProcesses-PT-10
BusinessProcesses-PT-11	BusinessProcesses-PT-12
BusinessProcesses-PT-13	BusinessProcesses-PT-14
BusinessProcesses-PT-15	BusinessProcesses-PT-16
BusinessProcesses-PT-17	BusinessProcesses-PT-18
BusinessProcesses-PT-19	BusinessProcesses-PT-20
CSRepetitions-COL-02	CSRepetitions-COL-03
CSRepetitions-COL-04	CSRepetitions-COL-05
CSRepetitions-COL-07	CSRepetitions-COL-10
CSRepetitions-PT-02	CSRepetitions-PT-03
CSRepetitions-PT-04	CSRepetitions-PT-05
CSRepetitions-PT-07	CSRepetitions-PT-10
CircadianClock-PT-000001	CircadianClock-PT-000010
CircadianClock-PT-000100	CircadianClock-PT-001000
CircadianClock-PT-010000	CircadianClock-PT-100000
CircularTrains-PT-012	CircularTrains-PT-024
CircularTrains-PT-048	CircularTrains-PT-096
CircularTrains-PT-192	CircularTrains-PT-384
CircularTrains-PT-768	ClientsAndServers-PT-N0001P0
ClientsAndServers-PT-N0002P0	ClientsAndServers-PT-N0002P1
ClientsAndServers-PT-N0005P0	ClientsAndServers-PT-N0005P1
ClientsAndServers-PT-N0010P0	ClientsAndServers-PT-N0010P1
ClientsAndServers-PT-N0010P2	ClientsAndServers-PT-N0020P0
ClientsAndServers-PT-N0020P1	ClientsAndServers-PT-N0020P2
ClientsAndServers-PT-N0020P3	ClientsAndServers-PT-N0020P4
ClientsAndServers-PT-N0050P0	ClientsAndServers-PT-N0100P0
ClientsAndServers-PT-N0200P0	ClientsAndServers-PT-N0500P0
ClientsAndServers-PT-N1000P0	ClientsAndServers-PT-N2000P0

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
ClientsAndServers-PT-N5000P0	CloudDeployment-PT-2a
CloudDeployment-PT-2b	CloudDeployment-PT-3a
CloudDeployment-PT-3b	CloudDeployment-PT-4a
CloudDeployment-PT-4b	CloudDeployment-PT-5a
CloudDeployment-PT-5b	CloudDeployment-PT-6a
CloudDeployment-PT-6b	CloudDeployment-PT-7a
CloudDeployment-PT-7b	CloudOpsManagement-PT-00002by00001
CloudOpsManagement-PT-00005by00002	CloudOpsManagement-PT-00010by00005
CloudOpsManagement-PT-00020by00010	CloudOpsManagement-PT-00040by00020
CloudOpsManagement-PT-00080by00040	CloudOpsManagement-PT-00160by00080
CloudOpsManagement-PT-00320by00160	CloudOpsManagement-PT-00640by00320
CloudOpsManagement-PT-01280by00640	CloudOpsManagement-PT-02560by01280
CloudOpsManagement-PT-05120by02560	CloudOpsManagement-PT-10240by05120
CloudOpsManagement-PT-20480by10240	CloudReconfiguration-PT-301
CloudReconfiguration-PT-302	CloudReconfiguration-PT-303
CloudReconfiguration-PT-304	CloudReconfiguration-PT-305
CloudReconfiguration-PT-306	CloudReconfiguration-PT-307
CloudReconfiguration-PT-308	CloudReconfiguration-PT-309
CloudReconfiguration-PT-310	CloudReconfiguration-PT-311
CloudReconfiguration-PT-312	CloudReconfiguration-PT-313
CloudReconfiguration-PT-314	CloudReconfiguration-PT-315
CloudReconfiguration-PT-316	CloudReconfiguration-PT-317
CloudReconfiguration-PT-318	CloudReconfiguration-PT-319
CloudReconfiguration-PT-320	CloudReconfiguration-PT-401
CloudReconfiguration-PT-402	DES-PT-00a
DES-PT-00b	DES-PT-01a
DES-PT-01b	DES-PT-02a
DES-PT-02b	DES-PT-05a
DES-PT-05b	DES-PT-10a
DES-PT-10b	DES-PT-20a
DES-PT-20b	DES-PT-30a
DES-PT-30b	DES-PT-40a
DES-PT-40b	DES-PT-50a
DES-PT-50b	DES-PT-60a
DES-PT-60b	DLCflexbar-PT-2a
DLCflexbar-PT-2b	DLCflexbar-PT-3a
DLCflexbar-PT-3b	DLCflexbar-PT-4a
DLCflexbar-PT-4b	DLCflexbar-PT-5a
DLCflexbar-PT-5b	DLCflexbar-PT-6a
DLCflexbar-PT-6b	DLCflexbar-PT-7a
DLCflexbar-PT-7b	DLCflexbar-PT-8a
DLCflexbar-PT-8b	DLCround-PT-03a
DLCround-PT-03b	DLCround-PT-04a
DLCround-PT-04b	DLCround-PT-05a
DLCround-PT-05b	DLCround-PT-06a
DLCround-PT-06b	DLCround-PT-07a
DLCround-PT-07b	DLCround-PT-08a
DLCround-PT-08b	DLCround-PT-09a
DLCround-PT-09b	DLCround-PT-10a

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
DLCround-PT-10b	DLCround-PT-11a
DLCround-PT-11b	DLCround-PT-12a
DLCround-PT-12b	DLCround-PT-13a
DLCround-PT-13b	DLCshifumi-PT-2a
DLCshifumi-PT-2b	DLCshifumi-PT-3a
DLCshifumi-PT-3b	DLCshifumi-PT-4a
DLCshifumi-PT-4b	DLCshifumi-PT-5a
DLCshifumi-PT-5b	DLCshifumi-PT-6a
DLCshifumi-PT-6b	DNAwalker-PT-01track12Block1
DNAwalker-PT-02track12Block2	DNAwalker-PT-03track12BlockBoth
DNAwalker-PT-04track28LL	DNAwalker-PT-05track28LR
DNAwalker-PT-06track28RL	DNAwalker-PT-07track28RR
DNAwalker-PT-08ringLL	DNAwalker-PT-09ringLR
DNAwalker-PT-10ringRL	DNAwalker-PT-11ringRR
DNAwalker-PT-12ringLLLarge	DNAwalker-PT-13ringRLLarge
DNAwalker-PT-14ringLRLarge	DNAwalker-PT-15ringRRLarge
DNAwalker-PT-16redondantChoiceR	DNAwalker-PT-17redondantChoiceL
DNAwalker-PT-18lozangeBlock	DatabaseWithMutex-COL-02
DatabaseWithMutex-COL-04	DatabaseWithMutex-COL-10
DatabaseWithMutex-COL-20	DatabaseWithMutex-COL-40
DatabaseWithMutex-PT-02	DatabaseWithMutex-PT-04
DatabaseWithMutex-PT-10	DatabaseWithMutex-PT-20
DatabaseWithMutex-PT-40	Dekker-PT-010
Dekker-PT-015	Dekker-PT-020
Dekker-PT-050	Dekker-PT-100
Dekker-PT-200	Diffusion2D-PT-D05N010
Diffusion2D-PT-D05N050	Diffusion2D-PT-D05N100
Diffusion2D-PT-D05N150	Diffusion2D-PT-D05N200
Diffusion2D-PT-D05N250	Diffusion2D-PT-D05N300
Diffusion2D-PT-D05N350	Diffusion2D-PT-D10N010
Diffusion2D-PT-D10N050	Diffusion2D-PT-D10N100
Diffusion2D-PT-D10N150	Diffusion2D-PT-D10N200
Diffusion2D-PT-D20N010	Diffusion2D-PT-D20N050
Diffusion2D-PT-D20N100	Diffusion2D-PT-D20N150
Diffusion2D-PT-D30N010	Diffusion2D-PT-D30N050
Diffusion2D-PT-D30N100	Diffusion2D-PT-D30N150
Diffusion2D-PT-D40N010	Diffusion2D-PT-D40N050
Diffusion2D-PT-D40N100	Diffusion2D-PT-D40N150
Diffusion2D-PT-D50N010	Diffusion2D-PT-D50N050
Diffusion2D-PT-D50N100	Diffusion2D-PT-D50N150
DiscoveryGPU-PT-06a	DiscoveryGPU-PT-06b
DiscoveryGPU-PT-07a	DiscoveryGPU-PT-07b
DiscoveryGPU-PT-08a	DiscoveryGPU-PT-08b
DiscoveryGPU-PT-09a	DiscoveryGPU-PT-09b
DiscoveryGPU-PT-10a	DiscoveryGPU-PT-10b
DiscoveryGPU-PT-11a	DiscoveryGPU-PT-11b
DiscoveryGPU-PT-12a	DiscoveryGPU-PT-12b
DiscoveryGPU-PT-13a	DiscoveryGPU-PT-13b
DiscoveryGPU-PT-14a	DiscoveryGPU-PT-14b

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
DiscoveryGPU-PT-15a	DiscoveryGPU-PT-15b
DotAndBoxes-COL-2	DotAndBoxes-COL-3
DotAndBoxes-COL-4	DotAndBoxes-COL-5
DoubleExponent-PT-001	DoubleExponent-PT-002
DoubleExponent-PT-003	DoubleExponent-PT-004
DoubleExponent-PT-010	DoubleExponent-PT-020
DoubleExponent-PT-100	DoubleExponent-PT-200
DrinkVendingMachine-COL-02	DrinkVendingMachine-COL-10
DrinkVendingMachine-COL-16	DrinkVendingMachine-COL-24
DrinkVendingMachine-COL-48	DrinkVendingMachine-COL-76
DrinkVendingMachine-COL-98	DrinkVendingMachine-PT-02
DrinkVendingMachine-PT-10	EGFr-PT-02010
EGFr-PT-10420	EGFr-PT-10421
ERK-PT-000001	ERK-PT-000010
ERK-PT-000100	ERK-PT-001000
ERK-PT-010000	ERK-PT-100000
Echo-PT-d02r09	Echo-PT-d02r11
Echo-PT-d02r15	Echo-PT-d02r19
Echo-PT-d03r03	Echo-PT-d03r05
Echo-PT-d03r07	Echo-PT-d04r03
Echo-PT-d05r03	EnergyBus-PT-none
Eratosthenes-PT-010	Eratosthenes-PT-020
Eratosthenes-PT-050	Eratosthenes-PT-100
Eratosthenes-PT-200	Eratosthenes-PT-500
FMS-PT-00002	FMS-PT-00005
FMS-PT-00010	FMS-PT-00020
FMS-PT-00050	FMS-PT-00100
FMS-PT-00200	FMS-PT-00500
FMS-PT-01000	FMS-PT-02000
FMS-PT-05000	FMS-PT-10000
FMS-PT-20000	FMS-PT-50000
FamilyReunion-COL-L00010M0001C001P001G001	FamilyReunion-COL-L00020M0002C001P001G001
FamilyReunion-COL-L00050M0005C002P002G001	FamilyReunion-COL-L00100M0010C005P005G002
FamilyReunion-COL-L00200M0020C010P010G005	FamilyReunion-COL-L00400M0040C020P020G001
FamilyReunion-COL-L00800M0080C040P040G020	FamilyReunion-COL-L01200M0120C060P060G030
FamilyReunion-COL-L03000M0300G150P150G075	FamilyReunion-COL-L05000M0500C250P250G125
FamilyReunion-COL-L08000M0800C400P400G200	FamilyReunion-COL-L12000M1200C600P600G300
FamilyReunion-PT-L00010M0001C001P001G001	FamilyReunion-PT-L00020M0002C001P001G001
FamilyReunion-PT-L00050M0005C002P002G001	FamilyReunion-PT-L00100M0010C005P005G002
FamilyReunion-PT-L00200M0020C010P010G005	FamilyReunion-PT-L00400M0040C020P020G001
FlexibleBarrier-PT-04a	FlexibleBarrier-PT-04b
FlexibleBarrier-PT-06a	FlexibleBarrier-PT-06b
FlexibleBarrier-PT-08a	FlexibleBarrier-PT-08b
FlexibleBarrier-PT-10a	FlexibleBarrier-PT-10b
FlexibleBarrier-PT-12a	FlexibleBarrier-PT-12b
FlexibleBarrier-PT-14a	FlexibleBarrier-PT-14b
FlexibleBarrier-PT-16a	FlexibleBarrier-PT-16b
FlexibleBarrier-PT-18a	FlexibleBarrier-PT-18b
FlexibleBarrier-PT-20a	FlexibleBarrier-PT-20b

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
FlexibleBarrier-PT-22a	FlexibleBarrier-PT-22b
GPPP-PT-C0001N0000000001	GPPP-PT-C0001N0000000010
GPPP-PT-C0001N0000000100	GPPP-PT-C0001N0000001000
GPPP-PT-C0001N0000010000	GPPP-PT-C0001N0000100000
GPPP-PT-C0010N0000000010	GPPP-PT-C0010N0000000100
GPPP-PT-C0010N1000000000	GPPP-PT-C0100N0000000010
GPPP-PT-C0100N0000000100	GPPP-PT-C0100N0000001000
GPPP-PT-C0100N0000010000	GPPP-PT-C0100N0000100000
GPPP-PT-C1000N0000000010	GPPP-PT-C1000N0000000100
GPPP-PT-C1000N0000001000	GlobalResAllocation-COL-03
GlobalResAllocation-COL-05	GlobalResAllocation-COL-06
GlobalResAllocation-COL-07	GlobalResAllocation-COL-09
GlobalResAllocation-COL-10	GlobalResAllocation-COL-11
GlobalResAllocation-PT-03	GlobalResAllocation-PT-05
HexagonalGrid-PT-110	HexagonalGrid-PT-126
HexagonalGrid-PT-226	HexagonalGrid-PT-316
HexagonalGrid-PT-410	HexagonalGrid-PT-516
HexagonalGrid-PT-816	HospitalTriage-PT-none
HouseConstruction-PT-00002	HouseConstruction-PT-00005
HouseConstruction-PT-00010	HouseConstruction-PT-00020
HouseConstruction-PT-00050	HouseConstruction-PT-00100
HouseConstruction-PT-00200	HouseConstruction-PT-00500
HouseConstruction-PT-01000	HouseConstruction-PT-02000
HouseConstruction-PT-04000	HouseConstruction-PT-08000
HouseConstruction-PT-16000	HouseConstruction-PT-32000
HypercubeGrid-PT-C3K4P4B12	HypercubeGrid-PT-C4K3P3B12
HypercubeGrid-PT-C5K3P3B15	HypertorusGrid-PT-d2k1p8b00
HypertorusGrid-PT-d2k2p1b00	HypertorusGrid-PT-d2k3p2b04
HypertorusGrid-PT-d3k3p2b06	HypertorusGrid-PT-d4k3p2b08
HypertorusGrid-PT-d5k3p2b10	IBM319-PT-none
IBM5964-PT-none	IBM703-PT-none
IBMB2S565S3960-PT-none	IOTPpurchase-PT-C01M01P01D01
IOTPpurchase-PT-C03M03P03D03	IOTPpurchase-PT-C05M04P03D02
IOTPpurchase-PT-C12M10P15D17	JoinFreeModules-PT-0003
JoinFreeModules-PT-0004	JoinFreeModules-PT-0005
JoinFreeModules-PT-0010	JoinFreeModules-PT-0020
JoinFreeModules-PT-0050	JoinFreeModules-PT-0100
JoinFreeModules-PT-0200	JoinFreeModules-PT-0500
JoinFreeModules-PT-1000	JoinFreeModules-PT-2000
JoinFreeModules-PT-5000	Kanban-PT-00005
Kanban-PT-00010	Kanban-PT-00020
Kanban-PT-00050	Kanban-PT-00100
Kanban-PT-00200	Kanban-PT-00500
Kanban-PT-01000	Kanban-PT-02000
Kanban-PT-05000	Kanban-PT-10000
Kanban-PT-20000	Kanban-PT-50000
LamportFastMutEx-COL-2	LamportFastMutEx-COL-3
LamportFastMutEx-COL-4	LamportFastMutEx-COL-5
LamportFastMutEx-COL-6	LamportFastMutEx-COL-7

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
LamportFastMutEx-COL-8	LamportFastMutEx-PT-2
LamportFastMutEx-PT-3	LamportFastMutEx-PT-4
LamportFastMutEx-PT-5	LamportFastMutEx-PT-6
LamportFastMutEx-PT-7	LamportFastMutEx-PT-8
MAPK-PT-00008	MAPK-PT-00020
MAPK-PT-00040	MAPK-PT-00080
MAPK-PT-00160	MAPK-PT-00320
MAPK-PT-00640	MAPK-PT-01280
MAPK-PT-02560	MAPK-PT-05120
MAPK-PT-10240	MAPKbis-PT-5310
MAPKbis-PT-5320	MultiwaySync-PT-none
NQueens-PT-05	NQueens-PT-08
NQueens-PT-10	NQueens-PT-12
NQueens-PT-15	NQueens-PT-20
NQueens-PT-25	NQueens-PT-30
NeighborGrid-PT-d2n3m1c12	NeighborGrid-PT-d2n3m1t12
NeighborGrid-PT-d3n3m1t11	NeighborGrid-PT-d4n3m2c23
NeighborGrid-PT-d5n4m1t35	NeoElection-COL-2
NeoElection-COL-3	NeoElection-COL-4
NeoElection-COL-5	NeoElection-COL-6
NeoElection-COL-7	NeoElection-COL-8
NeoElection-PT-2	NeoElection-PT-3
NeoElection-PT-4	NeoElection-PT-5
NeoElection-PT-6	NeoElection-PT-7
NeoElection-PT-8	NoC3x3-PT-1A
NoC3x3-PT-1B	NoC3x3-PT-2A
NoC3x3-PT-2B	NoC3x3-PT-3A
NoC3x3-PT-3B	NoC3x3-PT-4A
NoC3x3-PT-4B	NoC3x3-PT-5A
NoC3x3-PT-5B	NoC3x3-PT-6A
NoC3x3-PT-6B	NoC3x3-PT-7A
NoC3x3-PT-7B	NoC3x3-PT-8A
NoC3x3-PT-8B	PaceMaker-PT-none
ParamProductionCell-PT-0	ParamProductionCell-PT-1
ParamProductionCell-PT-2	ParamProductionCell-PT-3
ParamProductionCell-PT-4	ParamProductionCell-PT-5
Parking-PT-104	Parking-PT-208
Parking-PT-416	Parking-PT-432
Parking-PT-832	Parking-PT-864
PermAdmissibility-COL-01	PermAdmissibility-COL-02
PermAdmissibility-COL-05	PermAdmissibility-COL-10
PermAdmissibility-COL-20	PermAdmissibility-COL-50
PermAdmissibility-PT-01	PermAdmissibility-PT-02
PermAdmissibility-PT-05	PermAdmissibility-PT-10
PermAdmissibility-PT-20	PermAdmissibility-PT-50
Peterson-COL-2	Peterson-COL-3
Peterson-COL-4	Peterson-COL-5
Peterson-COL-6	Peterson-COL-7
Peterson-PT-2	Peterson-PT-3

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
Peterson-PT-4	Peterson-PT-5
Peterson-PT-6	Peterson-PT-7
PhaseVariation-PT-D02CS010	PhaseVariation-PT-D02CS100
PhaseVariation-PT-D05CS010	PhaseVariation-PT-D05CS100
PhaseVariation-PT-D10CS010	PhaseVariation-PT-D10CS100
PhaseVariation-PT-D20CS010	PhaseVariation-PT-D20CS100
PhaseVariation-PT-D30CS010	PhaseVariation-PT-D30CS100
Philosophers-COL-000005	Philosophers-COL-000010
Philosophers-COL-000020	Philosophers-COL-000050
Philosophers-COL-000100	Philosophers-COL-000200
Philosophers-COL-000500	Philosophers-COL-001000
Philosophers-COL-002000	Philosophers-COL-005000
Philosophers-COL-010000	Philosophers-COL-050000
Philosophers-COL-100000	Philosophers-PT-000005
Philosophers-PT-000010	Philosophers-PT-000020
Philosophers-PT-000050	Philosophers-PT-000100
Philosophers-PT-000200	Philosophers-PT-000500
Philosophers-PT-001000	Philosophers-PT-002000
Philosophers-PT-005000	Philosophers-PT-010000
PhilosophersDyn-COL-03	PhilosophersDyn-COL-10
PhilosophersDyn-COL-20	PhilosophersDyn-COL-50
PhilosophersDyn-COL-80	PhilosophersDyn-PT-03
PhilosophersDyn-PT-10	PhilosophersDyn-PT-20
Planning-PT-none	PolyORBLF-COL-S02J04T06
PolyORBLF-COL-S02J04T08	PolyORBLF-COL-S02J04T10
PolyORBLF-COL-S02J06T06	PolyORBLF-COL-S02J06T08
PolyORBLF-COL-S02J06T10	PolyORBLF-COL-S04J04T06
PolyORBLF-COL-S04J04T08	PolyORBLF-COL-S04J04T10
PolyORBLF-COL-S04J06T06	PolyORBLF-COL-S04J06T08
PolyORBLF-COL-S04J06T10	PolyORBLF-COL-S06J04T04
PolyORBLF-COL-S06J04T06	PolyORBLF-COL-S06J04T08
PolyORBLF-COL-S06J06T04	PolyORBLF-COL-S06J06T06
PolyORBLF-COL-S06J06T08	PolyORBLF-PT-S02J04T06
PolyORBLF-PT-S02J04T08	PolyORBLF-PT-S02J04T10
PolyORBLF-PT-S02J06T06	PolyORBLF-PT-S02J06T08
PolyORBLF-PT-S02J06T10	PolyORBLF-PT-S04J04T06
PolyORBLF-PT-S04J04T08	PolyORBLF-PT-S04J04T10
PolyORBLF-PT-S04J06T06	PolyORBLF-PT-S04J06T08
PolyORBLF-PT-S04J06T10	PolyORBLF-PT-S06J04T04
PolyORBLF-PT-S06J04T06	PolyORBLF-PT-S06J04T08
PolyORBLF-PT-S06J06T04	PolyORBLF-PT-S06J06T06
PolyORBLF-PT-S06J06T08	PolyORBNT-COL-S05J20
PolyORBNT-COL-S05J30	PolyORBNT-COL-S05J40
PolyORBNT-COL-S05J60	PolyORBNT-COL-S05J80
PolyORBNT-COL-S10J20	PolyORBNT-COL-S10J30
PolyORBNT-COL-S10J40	PolyORBNT-COL-S10J60
PolyORBNT-COL-S10J80	PolyORBNT-PT-S05J20
PolyORBNT-PT-S05J30	PolyORBNT-PT-S05J40
PolyORBNT-PT-S05J60	PolyORBNT-PT-S05J80

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
PolyORBNT-PT-S10J20	PolyORBNT-PT-S10J30
PolyORBNT-PT-S10J40	PolyORBNT-PT-S10J60
PolyORBNT-PT-S10J80	ProductionCell-PT-none
QuasiCertifProtocol-COL-02	QuasiCertifProtocol-COL-06
QuasiCertifProtocol-COL-10	QuasiCertifProtocol-COL-18
QuasiCertifProtocol-COL-22	QuasiCertifProtocol-COL-28
QuasiCertifProtocol-COL-32	QuasiCertifProtocol-PT-02
QuasiCertifProtocol-PT-06	QuasiCertifProtocol-PT-10
QuasiCertifProtocol-PT-18	QuasiCertifProtocol-PT-22
QuasiCertifProtocol-PT-28	QuasiCertifProtocol-PT-32
RERS17pb113-PT-1	RERS17pb113-PT-2
RERS17pb113-PT-3	RERS17pb113-PT-4
RERS17pb113-PT-5	RERS17pb113-PT-6
RERS17pb113-PT-7	RERS17pb113-PT-8
RERS17pb113-PT-9	RERS17pb114-PT-1
RERS17pb114-PT-2	RERS17pb114-PT-3
RERS17pb114-PT-4	RERS17pb114-PT-5
RERS17pb114-PT-6	RERS17pb114-PT-7
RERS17pb114-PT-8	RERS17pb114-PT-9
RERS17pb115-PT-1	RERS17pb115-PT-2
RERS17pb115-PT-3	RERS17pb115-PT-4
RERS17pb115-PT-5	RERS17pb115-PT-6
RERS17pb115-PT-7	RERS17pb115-PT-8
RERS17pb115-PT-9	Raft-PT-02
Raft-PT-03	Raft-PT-04
Raft-PT-05	Raft-PT-06
Raft-PT-07	Raft-PT-08
Raft-PT-09	Raft-PT-10
Railroad-PT-005	Railroad-PT-010
Railroad-PT-020	Railroad-PT-050
Railroad-PT-100	Referendum-COL-0010
Referendum-COL-0015	Referendum-COL-0020
Referendum-COL-0050	Referendum-COL-0100
Referendum-COL-0200	Referendum-COL-0500
Referendum-COL-1000	Referendum-PT-0010
Referendum-PT-0015	Referendum-PT-0020
Referendum-PT-0050	Referendum-PT-0100
Referendum-PT-0200	Referendum-PT-0500
Referendum-PT-1000	RefineWMG-PT-002002
RefineWMG-PT-002003	RefineWMG-PT-005005
RefineWMG-PT-005006	RefineWMG-PT-007007
RefineWMG-PT-007008	RefineWMG-PT-010010
RefineWMG-PT-010011	RefineWMG-PT-015015
RefineWMG-PT-015016	RefineWMG-PT-025025
RefineWMG-PT-025026	RefineWMG-PT-050050
RefineWMG-PT-050051	RefineWMG-PT-100100
RefineWMG-PT-100101	ResAllocation-PT-R002C002
ResAllocation-PT-R003C002	ResAllocation-PT-R003C003
ResAllocation-PT-R003C005	ResAllocation-PT-R003C010

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
ResAllocation-PT-R003C015	ResAllocation-PT-R003C020
ResAllocation-PT-R003C050	ResAllocation-PT-R003C100
ResAllocation-PT-R005C002	ResAllocation-PT-R010C002
ResAllocation-PT-R015C002	ResAllocation-PT-R020C002
ResAllocation-PT-R050C002	ResAllocation-PT-R100C002
Ring-PT-none	RobotManipulation-PT-00001
RobotManipulation-PT-00002	RobotManipulation-PT-00005
RobotManipulation-PT-00010	RobotManipulation-PT-00020
RobotManipulation-PT-00050	RobotManipulation-PT-00100
RobotManipulation-PT-00200	RobotManipulation-PT-00500
RobotManipulation-PT-01000	RobotManipulation-PT-02000
RobotManipulation-PT-05000	RobotManipulation-PT-10000
RwMutex-PT-r0010w0010	RwMutex-PT-r0010w0020
RwMutex-PT-r0010w0050	RwMutex-PT-r0010w0100
RwMutex-PT-r0010w0500	RwMutex-PT-r0010w1000
RwMutex-PT-r0010w2000	RwMutex-PT-r0020w0010
RwMutex-PT-r0100w0010	RwMutex-PT-r0500w0010
RwMutex-PT-r1000w0010	RwMutex-PT-r2000w0010
SafeBus-COL-03	SafeBus-COL-06
SafeBus-COL-10	SafeBus-COL-15
SafeBus-COL-20	SafeBus-COL-50
SafeBus-COL-80	SafeBus-PT-03
SafeBus-PT-06	SafeBus-PT-10
SafeBus-PT-15	SafeBus-PT-20
SharedMemory-COL-000005	SharedMemory-COL-000010
SharedMemory-COL-000020	SharedMemory-COL-000050
SharedMemory-COL-000100	SharedMemory-COL-000200
SharedMemory-COL-000500	SharedMemory-COL-001000
SharedMemory-COL-002000	SharedMemory-COL-005000
SharedMemory-COL-010000	SharedMemory-COL-020000
SharedMemory-COL-050000	SharedMemory-COL-100000
SharedMemory-PT-000005	SharedMemory-PT-000010
SharedMemory-PT-000020	SharedMemory-PT-000050
SharedMemory-PT-000100	SharedMemory-PT-000200
SimpleLoadBal-PT-02	SimpleLoadBal-PT-05
SimpleLoadBal-PT-10	SimpleLoadBal-PT-15
SimpleLoadBal-PT-20	SmallOperatingSystem-PT-MT0016DC0008
SmallOperatingSystem-PT-MT0032DC0008	SmallOperatingSystem-PT-MT0032DC0016
SmallOperatingSystem-PT-MT0064DC0016	SmallOperatingSystem-PT-MT0064DC0032
SmallOperatingSystem-PT-MT0128DC0032	SmallOperatingSystem-PT-MT0128DC0064
SmallOperatingSystem-PT-MT0256DC0064	SmallOperatingSystem-PT-MT0256DC0128
SmallOperatingSystem-PT-MT0512DC0128	SmallOperatingSystem-PT-MT0512DC0256
SmallOperatingSystem-PT-MT1024DC0256	SmallOperatingSystem-PT-MT1024DC0512
SmallOperatingSystem-PT-MT2048DC0512	SmallOperatingSystem-PT-MT2048DC1024
SmallOperatingSystem-PT-MT4096DC1024	SmallOperatingSystem-PT-MT4096DC2048
SmallOperatingSystem-PT-MT8192DC2048	SmallOperatingSystem-PT-MT8192DC4096
Solitaire-PT-EngCT7x7	Solitaire-PT-EngNC7x7
Solitaire-PT-FrnCT7x7	Solitaire-PT-FrnNC7x7
Solitaire-PT-SqrCT5x5	Solitaire-PT-SqrNC5x5

Table 3: names of all the model/instances of "known" models

Name of the model/instances ("known" models)	
SquareGrid-PT-020102	SquareGrid-PT-040204
SquareGrid-PT-080408	SquareGrid-PT-100510
SquareGrid-PT-130613	SwimmingPool-PT-01
SwimmingPool-PT-02	SwimmingPool-PT-03
SwimmingPool-PT-04	SwimmingPool-PT-05
SwimmingPool-PT-06	SwimmingPool-PT-07
SwimmingPool-PT-08	SwimmingPool-PT-09
SwimmingPool-PT-10	TCPcondis-PT-05
TCPcondis-PT-10	TCPcondis-PT-15
TCPcondis-PT-20	TCPcondis-PT-25
TCPcondis-PT-30	TCPcondis-PT-35
TCPcondis-PT-40	TCPcondis-PT-50
TokenRing-COL-005	TokenRing-COL-010
TokenRing-COL-015	TokenRing-COL-020
TokenRing-COL-030	TokenRing-COL-040
TokenRing-COL-050	TokenRing-COL-100
TokenRing-COL-200	TokenRing-COL-500
TokenRing-PT-005	TokenRing-PT-010
TokenRing-PT-015	TokenRing-PT-020
TokenRing-PT-030	TokenRing-PT-040
TokenRing-PT-050	TriangularGrid-PT-1200
TriangularGrid-PT-1500	TriangularGrid-PT-2011
TriangularGrid-PT-3011	TriangularGrid-PT-3026
TriangularGrid-PT-4022	TriangularGrid-PT-5020
TriangularGrid-PT-5046	UtahNoC-PT-none
Vasy2003-PT-none	VehicularWifi-COL-none

Table 3: names of all the model/instances of "known" models

B Appendix – An invocation example

We provide below an example of execution with our dummy tool. Execution was performed with the "personnal" *BenchKit* version as for 2018.

```
[fko TToolSubmissionKit]$ time ./BenchKitStart.sh mcc2015.vmdk StateSpace dummyTool PolyORBLF-COL-S02J04T06
   no memory confinment provided, assuming 1024 MBytes
   no VNC port specified, assuming 42
   no ssh redirection port specified, assuming 2222

execution on quadhexa-2.u-paris10.fr (runId=testing-run)
========================================================================
runnning dummyTool on PolyORBLF-COL-S02J04T06 (StateSpace)
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.
We got on stdout:
Probing ssh
Waiting ssh to respond
Ssh up and responding
========================================================================
 Generated by BenchKit version MCC2015 (monitoring deactivated, Feb 10, 2015)
    Executing tool dummyTool:
    Test is PolyORBLF-COL-S02J04T06, examination is StateSpace
========================================================================

--------------------
content from stdout:

START 1423653909
======================================================
== this is MyTool, a dummy example for the MCC'2015 ==
======================================================
Runing PolyORBLF (COL), instance S02J04T06

This tool just provides known information about state space of known
models for the first instance. This information comes from  the past
editions  of the  model checking contest  and is provided to let you
check that your tool provides appropriate results

STATE_SPACE STATES 104388 TECHNIQUES DUMMY_TECHNIQUE1 DUMMY_TECHNIQUE2
STATE_SPACE TRANSITIONS 193716 TECHNIQUES DUMMY_TECHNIQUE1
STATE_SPACE MAX_TOKEN_PER_MARKING -1 TECHNIQUES DUMMY_TECHNIQUE2
STATE_SPACE MAX_TOKEN_IN_PLACE -1 TECHNIQUES DUMMY_TECHNIQUE1 DUMMY_TECHNIQUE2
STOP 1423653909

--------------------
content from stderr:


--------------------
content from /tmp/BenchKit_head_log_file.1651:


real 0m23.504s
user 0m0.180s
sys  0m0.030s
```

The `--help` argument produces a small help as shown below

```
[fko  ./BenchKitStart.sh --help
usage: ./BenchKitStart.sh [-m <val>] [-vnc <val>] [-ssh <val>] <disk image> <bk-examination> <tool-name> <input>
       -m: <val> Mbyte of memory confinment are assumed (default is 1024)
     -vnc: <val> is the VNC port for the launched VM (default is 42)
```

```
     -ssh: <val> is the SSH port for the launched VM (default is 2222)

     <disk image>     : the path of the disk image to be booted and executed by the VM
     <bk-examination> : see BenchKit documentation, the variable defining what operation is to be executed on the VM
     <tool-name>      : see BenchKit documentation, the name of the tool
     <input>          : see BenchKit documentation, the name of the directory where the tool is executed

IMPORTANT: you must run ./BenchKitStart.sh in the directory you unpack the distribution.
IMPORTANT: in this version, only one core is allocated to the VM. This can be changed in the file vm.sh
           (at least for Qemu).
```

By default, only one core is allocated. To change this (QEMU only), have a look in the `launch_a_vm_with_qemu_or_vbox` in file `vm.sh`:

```
    $KVM -vnc :$VNC \
         -enable-kvm \
         -smp $SMP \
         -cpu Westmere
         -daemonize \
         -k $KEYBOARD \
         -m $MAXMEM \
         -hda $HDD \
         -net user,hostfwd=tcp:127.0.0.1:$SSHP-:22,restrict=yes \
         -net nic \
         -name MCC \
         -hdb $HDDINPUT
```

Please note that by default `$SMP` is set to 1 (if you use 4 cores, please change its value in the file `vm.sh`).

IMPORTANT: Please remind that the monitoring functions of *BenchKit* have been disabled to remove delicate dependencies and ease the installation on your target machine.