Monte Carlo Model Checker (MC2)

User Manual

Author: Robin Donaldson
Contact: mc2@brc.dcs.gla.ac.uk

Installation

This section describes the requirements and installation procedures for the MC2 tool and supporting software.

MC2 The MC2 model checking tool is written in Java, compiled using Java v1.5.0_07. You will need at least Java Runtime Environment v1.5 to run this tool. MC2 can be run by typing `java -jar MC2.jar` in the Unix shell.

BioNessie Please visit www.bionessie.com to download the latest version of the BioNessie software platform. BioNessie is the standard simulator used for the deterministic features of MC2.

gillespie2 We provide the source code for an altered version of gillespie2 v0.1.1. This should be compiled for your workstation's architecture and installed libraries by typing `make` then `make install` in the **src** directory. Please visit www.basis.ncl.ac.uk/software.html for library dependencies and further installation instructions. Note that the original gillespie2 v0.1.1 can be used with MC2, however the modified version provided is more suited for model checking.

Running the Simulators

BioNessie

Load the BioNessie simulator. Click File → Open File and locate an SBML model file. A new tab in the main frame should appear containing the model definition. Click the Simulate button to create the model simulation tab. Enter the simulation time period (Total Time) and number of output points (Steps), then click the Simulate button. Save this output to a file by clicking on the Data Output tab and clicking Save report.

gillespie2

You will wish to run the gillespie2 simulator at least 100 times to create a set of simulation runs for MC2. To do this, create a shell script **example.run** as follows:

```
#!/bin/sh

rm $1
for (( j = 0 ; j < 100; j++ ))
do
  ./gillespie2 -m example.xml -t 300 -p Prot1,Prot2,Prot3 >> $1
done
```

Where example.xml is the SBML model file you wish to simulate, 300 is the length of time to simulate and Prot1,Prot2,Prot3 is a comma-separated list of species to output. Run this file by typing `./example.run output _file` which will run the gillespie2 simulator 100 times creating and appending to **output_file**.

We have modified gillespie2 to become an exact Gillespie simulator outputting every molecular event in the simulation time. This provides more accurate input to the MC2 model checker. There is an added flag to the gillespie2 run command, `-p` which is followed by a comma-separated list of species which should be written to the output file – hence the output file will contain only concentration values and molecular events of these species. Finally, this version of gillespie2 outputs a blank line at the end of each simulation to aid the concatenation of many simulation outputs.

You may wish to simulate at a different number of molecules than have been provides in the **models** directory (e.g., Levchenko is supplied at 4 and 8 molecules however you may wish to simulate with 16). To do so, you should follow the "Rate constant conversion" chapter of Stochastic Modelling for Systems Biology by Darren J. Wilkinson (2006). Essentially for the Levchenko model, to multiply the initial concentrations by a factor the 2nd order equations must also be divided by this factor. However models containing different order equations need to be handled differently.

Introduction to PLTL

Properties in MC2 are expressed using a logic language called Probabilistic Linear-time Temporal Logic (PLTL). The basic element of this language is the Atomic Proposition (AP) which is part of a property without temporal operators. For example, the property "Protein1 is always greater than 10" contains an AP "Protein1 greater than 10". The syntax of an AP is:
$AP ::= AP \vee AP | AP \wedge AP | \neg AP | AP \rightarrow AP | value = value | value \neq value | value > value | value \geq value | value < value | value \leq value | true | false$

$value ::= value + value | value - value | value * value | value/value | [Species] | d[Species] | max([Species]) | Int | Real$

where *Species* is the name of any species in the simulation output, *Int* is any integer number and *Real* is any real number. [*Species*] is the species concentration, *d*[*Species*] is the species concentration derivative and *max*([*Species*]) is the maximum species concentration in the simulation output.

The overall PLTL definition is:
$\psi ::= P_{\trianglelefteq x}[\phi] | P_{\trianglelefteq x}[\phi\{AP\}]$

$\phi ::= X\phi | G\phi | F\phi | \phi U \phi | \phi R \phi | \phi \vee \phi | \phi \wedge \phi | \neg\phi | \phi \rightarrow \phi | AP$

$P_{\trianglelefteq x}$ is any inequality comparison of the probability of the property holding true, for example $P_{\geq 0.5}$. We also permit the expression $P_{=?}$ returning the value of the probability of the property holding true. We provide a choice in top-level PLTL such that a query of the form $\phi\{AP\}$ means ϕ is checked from the first time in the simulation output that AP is satisfied.

The temporal operators are defined as below:
Next (X) - The property must hold true in the next time point.
Globally (G) - The property must hold true always in the future.
Finally (F) - The property must hold true sometime in the future.
Until (U) - The first property must hold true until the second property holds true.
Release (R) - The second property can only ever not hold true if the first property becomes true.

Running MC2

Deterministic MC2

Deterministic MC2 takes as input a query definition file and simulation output file in the BioNessie format. The simulation output file should be generated either through BioNessie (described above) or through another simulator of your choice translated to the BioNessie output format. The query definition file must contain at least one query written in PLTL (one query per line) such as:

```
P>=1[ ([MEK_PP] < 0.001 ^ [ERK_PP] < 0.0002) U ([RafP] > 0.06) ]
P>=1[ ([RafP] > 0.06 ^ [ERK_PP] < 0.0002) => (([RafP] > 0.06 ^ [ERK_PP] < 0.0002) U
    ([MEK_PP] > 0.004)) ]
P>=1[ ([RafP] > 0.06 ^ [MEK_PP] > 0.004) => (([RafP] > 0.06 ^ [MEK_PP] > 0.004) U ([ERK_PP]
    > 0.0005)) ]
```

Run the following command in the MC2 directory:

```
java -jar MC2.jar det simulation_file query_file
```

The output from MC2 should be appear as follows:

```
true
false
true
```

Parameter Scan MC2

An example internal file from BioNessie has been included called **kholondenko_paramscan_Ki=1-30**. Model checking on this parameter scan output can be performed by running this command:

```
java -jar MC2.jar det kholondenko\_paramscan\_Ki=1-30 query_file
```

However, the parameter scan internal files are not easily accessible. MC2 model checking of parameter scans will be fully integrated in the next release of BioNessie.

Stochastic MC2

Stochastic MC2 takes as input a query definition file and simulation output file in a concatenation of the gillespie2 format. The simulation output file should be generated either through gillespie2 (described above) or through another simulator of your choice translated to the gillespie2 output format. The query definition file must contain at least one query written in PLTL (one query per line) such as:

```
P=?[ ([RafP] = 0) U ([RafP] > 0) {[RafP] = 0} ]
P=?[ ([RafP] = 1) U ([RafP] > 1) {[RafP] = 1} ]
P=?[ ([RafP] = 2) U ([RafP] > 2) {[RafP] = 2} ]
P=?[ ([RafP] = 3) U ([RafP] > 3) {[RafP] = 3} ]
P=?[ ([RafP] = 4) U ([RafP] > 4) {[RafP] = 4} ]
```

Unix scripts have been provided to create a query file for a query assessed at many levels. For example, to produce the query file above, run the following command in the **queries** directory, `./produceS1 4 1`. This will create the S1 query assessed up to level 4 in increments of 1 level.

Run the following command in the MC2 directory:

```
java -jar MC2.jar stoch simulation_file query_file
```

The output from MC2 should be appear as follows:

```
1.0,0.04,0.03,0.03,0.0
```

At greater numbers of molecules model checking may exhaust your workstation's memory. First it will be useful to increase the heap size in the Java Virtual Machine. This can be done by adding the initial and maximum heap size flags, `-Xms` and `-Xmx` respectively. Furthermore, MC2 accepts an optional parameter which is the maximum number of traces to hold in memory at one time. For example, when checking large traces you may wish to specify that the heap size is initially half and maximally the full size of the physical memory and only hold 1 trace in memory at any one time:

```
java -Xms512M -Xmx1024M -jar MC2.jar stoch simulation_file query_file 1
```

Distribution

MC2 v1.0 (17/01/08) ©Bioinformatics Research Centre, University of Glasgow is available at:
www.brc.dcs.gla.ac.uk/software/mc2

Simulation Output Format

Deterministic Format

```
<ignored lines>
```

```
----------------------------------------------------------------------
T|Protein1|Protein2|...|ProteinN
0.0|100|0|...|400
1.0|75|0|...|354
.
.
.
100.0|30|0|...|27
----------------------------------------------------------------------
<ignored lines>
```

Parameter Scan Format

The parameter scan format contains a comma-separated parameter list on the first line containing the parameter values used in the respective simulation outputs. The next lines are a concatenation of outputs in the deterministic format

```
1,2,...,10
<deterministic format>
<deterministic format>
.
.
.
<deterministic format>
```

Stochastic Format

The stochastic format is a concatenation (with a separating empty line) of many single simulation outputs in the format below:

```
Time Protein1 Protein2 ... ProteinN
0.0 100 0 ... 400
1.0 75 0 ... 354
.
.
.
100.0 30 0 ... 27
```