Macchiato
A Simple Petri Nets implementation for Python

Dr. Mark James Wootton
mark.wootton@nottingham.ac.uk

Tuesday 19th November, 2019

This document is current for Version-20190129. Note that is assumed that the reader is already familiar with the basics of standard Petri Net modelling [1].

1 Requirements

Python https://www.python.org – Essential

Macchiato has been tested with Python 2.7 and Python 3, and has the following library dependencies:

- `re`
- `os`
- `sys`
- `copy`
- `math`
- `time`
- `random`
- `subprocess`
- `collections`

Graphviz http://www.graphviz.org – Optional

Used for visualisation of Petri Nets; can read and convert `.dot` files to images.

2 Instructions

Macchiato Petri Net structures are stored in `.mpn` files. One may also create and manipulate Petri Net structures in a Python script, using the tools provided in the `PetriNet` module.

2.1 Macchiato Petri Net Files (`.mpn`) Files

`.mpn` files are use to store the structure of a Petri Net and the parameters for executing its simulation; blank lines and those beginning with "`#`" are ignored. They are loaded by the `read` function of the `Macchiato` module. Running the `Macchiato` module directly from the command line will execute a simulation of the Petri Net specified in the given `.mpn`, for example:

```
python Macchiato.py Example_PetriNet.mpn
```

An example of a short `.mpn` file may be found in section 2.1.7. By default, the number of simulations performed will be function of the `.mpn` file's parameters, but by adding an integer to the end of the command, this may be controlled manually. The command,

```
python Macchiato.py Example_PetriNet.mpn 5000
```

will execute 5000 simulation of `Example_PetriNet.mpn`. Minimal output will be written to the terminal to reduce runs times. If however, the user prefers to see the full output, verbose mode may be invoked by adding "`V`" as a final comand line argument.

2.1.1 Petri Net and Run Parameters

The first lines of the file should specify the basic parameters of the net:

`name` The label given to the Petri Net and used in output directories

`units` The units of time to be used by the Petri Net (Default is "hrs")

`runMode` The mode of integration to be used for simulation (Default is "`schedule`" – note: This is the only run mode that supports all types of transition timing and is actively maintained. Use other modes with caution.)

`dot` Write Petri Net state to `.dot` file (Default is "`False`")

`visualise` File format for visualisation conversion to image file. Supported types include, but are not limitted to, `png` and `pdf` (Default is "None", i.e. no conversion). Be aware that visualisation with significantly increase runtimes and is not advised for large simulation sets.

`details` Toggles label with Petri Net name, step, and clock in visualisations (Default is "`True`")

`useGroup` Toggles use of place and transition groups in visualisation (Default is "`True`")

`orientation` Option for visualisation. Options are "LR", "RL", "TB", and "BT".

`debug` Print debug (Semi-redundant, default is "`False`")

The next options parameterise the simulation.

`maxClock` Greatest clock duration permitted in any one simulation (Default is 10^6 units of time)

`maxSteps` Greatest number of steps permitted in anyone simulation (Default is 10^{12})

`simsFactor` Parametises the number of simulations conducted (Default is 1.5×10^3). Repetition of simulations ends once the total simulated time supasses the product of `maxClock` and `simsFactor`. If a set number of simulations is specified at the command line, `simsFactor` is overruled.

`history` Experimental feature: Collects and aggregates data if "`True`" (Default is "`False`")

`analysisStep` Time resolution of post-simulation analysis used if `History` feature is in use (Default is 100 units of time)

Parameters are specified by giving the name of the parameter, followed by a space, then the desired option.

2.1.2 Petri Net Structure

To indicate that one wishes to begin specifying places, include the line `Places`. On the following lines, specify the name of the place, and if a non-zero number of initial tokens is desired, include the number after the name, separated by a space. Similarly, to switch to specifying transitions, include the line `Transitions`. Transitions are specified by writing the name of the transition, its timing form, and the relevant parameters separated only by colons. To begin specifying incoming or outgoing arcs, leave a space and write "`IN`" or "`OUT`", as appropriate. Each connecting place is to be given after a space, with the weight option specified following its label, separated only by a colon (if no option is given, the arc weight defaults to 1). For example: `IN P1 P2:3 P3 OUT P4 P5:2` specifies three incoming arcs, the second of which has a weight of 3, and two outgoing arcs, the latter of which has a weight of 2.

2.1.3 Transition Timing options

Options for transition timings are as follows (use the order in which they are given here when specifying them):

`instant` No additional options

`rate` 1 option – Kinetic Monte Carlo style rate [2]

`uniform` 1 option – X where time, t, is uniformly distributed in $0 < t < X$

`delay` 1 option – Fixed delay of given duration

`weibull` 2 mandatory inputs, 1 optional – The Weibull distribution [3] is parameterised by η and β, such that $t = \eta[-\ln(X)]^{-\beta}$, and X is a random variable uniformly distributed in the range $0 < X < 1$. For $\beta = 1$, η is the mean duration, but otherwise these are not equal. Give the desired mean duration, $\langle t \rangle$, as the first variable and η will be generated by way of $\eta = \langle t \rangle \left[\Gamma\left(\beta^{-1} + 1\right)\right]^{-1}$. For example: `weibull:50000000:1.2` where $\langle t \rangle = 5{\times}10^7$, $\beta = 1.2$, and therefore $\eta = 5.315{\times}10^7$. The inclusion of the optional third input allows one to include a uncertainty quantification on the mean time to fire. If present, whenever a firing time is calculated, the value used for η is generated from a normal distribution [4] with the stored value of η as its mean, and the uncertainty parameter used as standard deviation, σ.

`lognorm` 2 options – Mean and standard deviation, μ and σ, for a Log-Normal Distribution [5].

`beta` 2 mandatory inputs, 1 optional. The Beta Distribution [6] is parameterised by α and β, and returns a value in the interval, $0 < t < 1$. A third parameter may be included as a multiplying factor, k, such that range is uniformly scaled to, $0 < t < k$.

`cyclic` 2 options – Frequency of firing (when the simulation clock is at an integer multiple) and offset.

A more detailed mathematical description of each distribution is given in the appendix.

2.1.4 Extended Arc Behaviours

Beyond the standard arcs found in basic Petri Net modelling, Macchiato also supports two additional options – the place conditional and the inhibit arc. These are respectively specified by adding the codes `:pnc` and `:inh` to the end of the arc specification, such as `P1:2:inh`. If the weight of an inhibit arc is satisfied by its place, it has the effect of preventing the connecting transition from firing, regardless of the state of any other connecting arcs. Place conditional arcs

are used to apply a modifier to a transition's probability distribution. Unlike standard or inhibit arcs they have no effect on whether a transition can fire. The calculation of the modifying factor and its application to each specific distribution is given in the appendix. Place conditional arcs are the only arc type that permits non-integer weights.

On visualisations, both of these arc types are drawn with dashed lines, blue in the case of place conditionals, and red in the case of inhibit arcs, terminating respectively in hollow and solid circles, instead of the arrow heads used for standard arcs.

2.1.5 Extended Transition Behaviours

To facilitate development of models for systems of high complexity, Macchiato includes two special behaviours which extend the feature-set of standard Petri Nets. Note that a transition may make both of these features concurrently.

`RESET` A reset transition has a list of places associated with it. If the transition fires, all places in the list are restored to their original token count, as specified at the beginning of the simulation. Reset transitions can be specified by the label "`RESET`", followed by a list places to reset, each separated by colons ("`RESET`" is separated from the list by a space).

`VOTE` At the of the transition description line, by adding the label "`VOTE`", mandatorily followed by a number separated by a space, the transition firing behaviour changes. Instead of requiring all incoming standard arcs to be satisfied, the transition instead will require only the quantity of arcs corresponding to the given number. Be aware that if the number of satisfied arcs is higher than this number, tokens will still be taken from all the corresponding places (no tokens will be taken from any place whose arc is not satisfied). Outgoing arcs from a vote transition behave normally. On visualisations, vote transitions are rendered as cuboids.

2.1.6 Visual Groupings

To assign a grouping for visualisation to a transition or place add the label "`GROUP`" to the end of the line on which it is specified followed by a space and an integer, which serves as its group assignment. Object in the same group will be displaced together if visualisation is active. Note that places and transitions have separate groupings, even if the same numbers are used.

2.1.7 Example `.mpn`

```
# Petri Net Parameters
        name Test
        units hrs
        runMode schedule
        visualise png
        dot True

# Run Parameters
        maxClock 1E3
        maxSteps 100
        simsFactor 1
        history False
        analysisStep 1

# Build Petri Net
Places
```

```
        P0 2
        P1
        P2
        P3

Transitions
        T0:lognorm:1:1 IN P0 OUT P1 P3
        T1:weibull:1:0.5 IN P1 OUT P2:2
        T2:delay:2 IN P2:2 P3:inh OUT P1
        T3:rate:15 IN P3:5:pcn P1 OUT P2
        R:cyclic:7:1 IN P2 RESET P0:P1:P3
        V:beta:1:2:0.25 IN P0 P1 P3 OUT P2 VOTE 2
```

2.2 Scripting with Macchiato

To create and manipulate Petri Nets via scripting, import `PetriNet` into a Python script. Documentation for the objects and functions required may be found in the modules itself. Note that if the `Macchiato` module is also imported, the `write` function may be used to export a created structure to an `.mpn` file. When called from within a script, `read` returns a `PetriNet` object. One may execute single simulations of a Petri Net by using the `PetriNet` object's `run` function.

3 Analysis Scripts

Two scripts are provided for basic analysis of the results. These are `TimingData.py` and `TransFireFrequency.py`. Note that the additional libraries `nunpy` and `matplotlib` are required.

3.1 `TimingData.py`

This script will provide information on the proportion of simulations ending in particular outcomes and the average durations of those sets, with standard error given. This is achieved by inspected of the final states of a given list of places. This list is specified by column numbers, which count from zero, and should be separated by colons, e.g. `3:8:16`. The script will also produce a histogram to represent the results, with *"Duration"* taking the same units as those specified in the simulated Petri Net. A plaintext file and an image are produced in the current working directory.

Example:

```
python TimingData.py Results_Folder 3:8:16
```

3.2 `TransFireFrequency.py`

This script produces statistics for transition firings, with standard error given. Simply provide the folder containing the results for inspection and a plaintext file will be produced in the current working directory.

Example:

```
python TransFireFrequency.py Results_Folder
```

Appendix

Scheduling Transition Firings & Governing Formulae

On the first simulation step following the satisfaction of the firing conditions of a timed transition, a firing time is scheduled using the appropriate governing formula from the list below. For the entirety of the time between the scheduling and the transition firing, the conditions must be met continuously. If a connected inhibit arc is satisfied, or if the requisite tokens are removed from the relevant places, the transition's firing will be removed from the schedule – *N.B.*: A transition upon whose firing both takes from, and adds to, the token count at a place does not interrupt the conditions for firing another transition dependant on the same place, providing the token count remains sufficient with respect the latter at the end of the simulation step. If the number of tokens feeding a connected place conditional arc changes, the scheduled time will be recalculated relative to the original time of token condition satisfaction. Should this produce a time in the past, the transition will be scheduled for the present simulation time. Instant transitions always have priority over timed transitions, even if the time until the later fires is zero. When multiple instant transitions are available to fire on a given simulation step, one will be chosen at random, with equal weighting given to those transitions.

In the following subsections, the time from a transition conditions being first fulfilled until it fires is given the symbol, t, where $t \in \mathbb{R}_{\geq 0}$.

A.1 Place Conditional Transitions

A modifying factor, P, is generated for place conditional transitions by equation (1).

$$P = 1 + \sum_i W_i N_i \tag{1}$$

where W_i is the weight of the place conditional arc and N_i is the number of tokens on the corresponding place. The specific manor in which P modifies transition timing is dependant on the specific timed transition type.

A.2 Fixed Delay

A timed transition with a fixed delay will fire after a set period, *i.e.* a transition with a fixed delay of five seconds whose firing conditions are met when the simulation clock at ten seconds will be scheduled to fire at fifteen seconds. If the delay is given by a, then the place conditional modification is given by $a \to \frac{a}{P}$. In terms of a, the probability density function is,

$$f(t;a) = \begin{cases} 1 & \text{for} \quad t = a \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

A.3 Uniform Distribution

The uniform distribution gives even weighting to values within a range, parametrised by u, giving the probability density function, $f(t;u)$, where:

$$f(t;u) = \begin{cases} \frac{1}{u} & \text{for} \quad t \in (0,u] \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

with the place conditional modification being given by $u \to \frac{u}{P}$.

A.4 Cyclic Distribution

A timed transition with the cyclic distribution, parametrised by c and ω, will fire the next time that the simulation clock is equal to an integer multiple of c, with the caveat that a cyclic distribution transition prohibited from firing when the clock is at zero, and also that an offset, ω may be applied. If the simulation time at which the transition's conditions are satisfied is T, then probability density function, $f(t; c, \omega, T)$, is,

$$f(t; c, \omega, T) = -T + \begin{cases} 1 & \text{for} \quad t = \begin{cases} T - \omega + z_1 & \text{for} \quad T - \omega + z_1 > 0 \\ T - \omega + z_2 & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where z_1 is the smallest non-negative number for which $T - \omega + z_1$ is an integer multiple of c and such that $T + t \leq 0$ and that $T + t$ is not the last time at which the transition fired, and where $z_2 = z_1 + c$. The place conditional effect on c is $c \to \frac{c}{P}$, with ω unchanging.

A.5 Weibull Distribution

The Weibull Distribution [3] is used widely in reliability engineering. It is formed from a scale parameter, η, related to the mean time to failure, $\langle t \rangle$, by equation (5), and a shape parameter, β, which weighs the distribution towards early or later failures [7].

$$\eta = \frac{\langle t \rangle}{\Gamma\left(\beta^{-1} + 1\right)} \tag{5}$$

The Gamma Function [8] is defined by:

$$\Gamma(x) = \int_0^\infty \left(z^{x-1} \exp\left[-z\right]\right) dz \tag{6}$$

For $\beta < 1$, failure rate decreases with time, giving systems a high initial failure probability. With $\beta > 1$, the failure grows with time, representative of an ageing process. If $\beta = 1$, the failure rate is unchanging across time. In equation (7), the probability density function, $f(t; \eta, \beta)$, of the Weibull Distribution is given by:

$$f(t; \eta, \beta) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} \exp\left(-\left[\frac{t}{\eta}\right]^{\beta}\right) \tag{7}$$

The place conditional effect for Weibull distributions is $\eta \to \frac{\eta}{P}$

A.6 Log-Normal Distribution

The Log-Normal Distribution is given by the result of the exponential function applied to the result of a normal distribution [4], such that the resulting variable is normally distributed in magnitude [5]. As seen in equation (8), for a normal distribution with mean and standard deviation given by μ and σ, the probability density function, $f(t; \mu, \sigma)$, is:

$$f(t; \mu, \sigma) = \frac{1}{t\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left[\frac{\ln(t) - \mu}{\sigma}\right]^2\right) \tag{8}$$

Place conditional timed transitions with Log-Normal distributions are modified by $\mu \to \frac{\mu}{P}$.

A.7 Beta Distribution

The Beta Distribution [6] appears frequently in work involving Bayesian Networks and takes two shape parameters, α and β. Macchiato allows one to scale the result (normally between 0 and 1) by a factor, k, giving the probability density function, $f(t; \alpha, \beta, k)$:

$$f(t;\alpha,\beta,k) = \begin{cases} \left(\left[\frac{x}{k}\right]^{\alpha-1}\left[1-\frac{x}{k}\right]^{\beta-1}\right)\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} & \text{for} \quad t \in (0,k) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

with the Gamma Function [8] as defined in equation (6). The place conditional modification for Beta Distributions is $k \to \frac{k}{P}$.

A.8 KMC Rate Distribution

`Macchiato` also supports a distribution based on the KMC algorithm [2], parametrised by a rate, r. It has the probability density function of this distribution, $f(t; r)$, given by,

$$f(t;r) = r\exp(-rt) \tag{10}$$

The place conditional effect for this distribution is $r \to rP$.

References

[1] Carl Adam Petri. *Kommunikation mit Automaten (In German).* PhD thesis, Technical University Darmstadt, 1962.

[2] Arthur F. Voter. Introduction to the Kinetic Monte Carlo Method. *K.E. Sickafus et al. (eds.), Radiation Effects in Solids*, pages 1–23, 2007.

[3] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes.* McGraw Hill, 4th edition, 2002.

[4] Eric W. Weisstein. "Normal Distribution." From MathWorld--A Wolfram Web Resource. *mathworld.wolfram.com/NormalDistribution.html Accessed March 2019*, Last edited: 2019.

[5] Brian Dennis and G. P. Patil. *Lognormal Distributions, Theory and Applications.* Marcel Dekker New York, 1987.

[6] Arjun K. Gupta and Saralees Nadarajah, editors. *Handbook of Beta Distribution and Its Applications.* Marcel Dekker, Inc, 2004.

[7] R. Jiang and D. N. P. Murthy. A study of Weibull shape parameter: Properties and significance. *Reliability Engineering and System Safety*, 96:1619–1626, 2011.

[8] Eric W. Weisstein. "Gamma Function." From MathWorld--A Wolfram Web Resource. *www.mathworld.wolfram.com/GammaFunction.html Accessed September 2018*, Last edited: 2005.