High-level stochastic simulation for exploring the dynamics of biological systems

1st George Assaf
Brandenburg University of Technology
Postbox 101344, 03013 Cottbus, Germany
george.assaf@b-tu.de

2nd Fei Liu
South China University of Technology
Guangzhou 510006, P.R. China
feiliu@scut.edu.cn

3rd Monika Heiner
Brandenburg University of Technology
Postbox 101344, 03013 Cottbus, Germany
monika.heiner@b-tu.de

***Abstract*—We propose a new algorithm for exploring behaviours of biological systems with repeated structures modelled as coloured stochastic Petri nets. Current approaches executing coloured stochastic Petri net models over time require unfolding the model at hand into its uncoloured counterpart as a preliminary step before performing the actual stochastic simulation. Unfolding coloured models with large scaling factors may lead to a high memory demand, as high-level components need to be flattened into their basic modelling elements, biologically interpreted as reactions and their associated species. To address this issue, it is crucial to simulate the model on-the-fly without having to perform model unfolding. Our proposed algorithm incorporates the colouring concept of Petri nets into the well-known *Gillespie's* stochastic simulation algorithm. The algorithm is implemented in the powerful Petri nets modelling and simulation tool *Snoopy*. In this paper, we present two biological case studies, for which the new algorithm reproduced results identical to the unfolding approach.**

***Index Terms*—biological systems with repeated structures, model analysis, stochastic simulation, coloured stochastic Petri nets**

I. Introduction

A. Modelling biological systems with repeated structures

Over the last decade, modelling and analysing biological systems with similar structures have become one of the focal points in the computational modelling community [1]–[4]. This interest stems from the fact that biological systems comprise thousands of similar biochemical reactions and associated species, that need to be modelled in a convenient manner for the purpose of predicting the behaviour of the modelled system. Coloured stochastic Petri nets ($\mathcal{SPN}^{\mathcal{C}}$) are an excellent modelling paradigm for obtaining quantitative insights into the inherent stochasticity of the modelled system, as they come with various appropriate modelling elements such as discrete places and stochastic transitions.

Petri nets ($\mathcal{PN}$) [5] have increasingly become an attractive formalism for modelling biological systems as they offer various graphical modelling elements, which can be read in the stochastic, deterministic, and hybrid paradigms for quantitatively analysing the behaviour of biological systems, see [6]–[8]. The $\mathcal{PN}$ modelling elements basically comprise places, transitions, and directed arcs for connecting places and transitions. In the context of biological systems, places model species whose molecule number/concentration (depending on the modelling paradigm) is represented by the number of tokens (also called marking) that are held by the places, whereas transitions model the system reactions, and the connecting arcs together with their associated arc weights represent the number of molecules of each involved reactant/product. Moreover, the occurrence of a reaction is equivalent to firing an enabled transition, for which all the transition's pre-places hold a number of tokens greater than or equal to the connected arc weights. Firing a transition results in consuming a number of tokens (from pre-places) equal to the corresponding arc weights, while the post-places are updated by adding tokens equal to the connected arc weights. This family of Petri nets are called uncoloured quantitative Petri nets comprising stochastic, continuous and hybrid Petri nets. For more insights into adopting $\mathcal{PN}$ for modelling and simulating biological systems, we recommend to consult further material such as [9], [10].

Based on the traditional $\mathcal{PN}$, coloured Petri nets ($\mathcal{PN}^{\mathcal{C}}$) [11]–[13] have been introduced offering the concept of colour sets. A colour set is a user-defined data type with a finite number of elements (colours) that can be assigned to the net places to represent several distinguishable copies of the modelled object. This means that each token will be differentiated by a unique colour, which has the advantage of representing similar objects using one coloured place rather than having several places for each object. For instance, to encode 100 genes, a colour set, e.g. *GeneSet*, of *integer* type with 100 elements (colours) is defined, and then later assigned to one coloured place (rather than having 100 places, which would be the case for an uncoloured $\mathcal{PN}$). It is worth mentioning that colour sets are also used to define user declarations

George Assaf was supported by the *Katholischer Akademischer Ausländer-Dienst* (KAAD) scholarship. Fei Liu is supported by the National Natural Science Foundation of China (62273153).

such as variables and functions, in the very same way as in programming languages, which can be deployed later for annotating places, transitions, and arcs to obtain a concise net structure of the modelled system. Therefore, colouring propound major advantages for expressing complex systems in a compact and scaleable fashion. Like plain (uncoloured) Petri nets, $\mathcal{PN}^{\mathcal{C}}$ offer quantitative classes for exploring the behaviour of biological systems with similar structures, see for example [2], [4], [14].

In this paper, we are interested in analysing the behaviours of biological systems with similar structures modelled using coloured stochastic Petri nets. Stochastic simulation is able to capture a realistic representation of random fluctuations at the molecular level and its effects on the behaviour of the modelled system. Notably, we propose a novel algorithm for analysing biological systems with similar structures modelled as $\mathcal{SPN}^{\mathcal{C}}$, as former well established approaches require unfolding an $\mathcal{SPN}^{\mathcal{C}}$ model into its plain counterpart as a mandatory step before performing the actual simulation [15]. However, model unfolding consumes memory space, as each coloured modelling element, e.g. coloured places, has to be flattened into uncoloured elements in the corresponding equivalent uncoloured Petri net. Unfolding may become a significant concern and limiting factor, especially in the case of $\mathcal{SPN}^{\mathcal{C}}$ models involving substantial scaling factors [1], [15]–[17]. Moreover, the unfolded model has to be fed to the simulator which will lead to additional overhead on both memory consumption and computational resources.

B. Coloured stochastic Petri nets

Coloured stochastic Petri nets [12], [18], [19] represent the coloured version of standard stochastic Petri nets ($\mathcal{SPN}$) [20], [21]. $\mathcal{SPN}^{\mathcal{C}}$ serve as a powerful tool for modelling and analysing biological systems with analogous structures. They are able to capture the system's behaviour that is not observable in deterministic approaches by considering random interactions and fluctuations that are induced by the modelled system, and thus enable a much deeper understanding of the underlying behaviour of biological phenomena, see for example [1], [2], [18], [19].

In plain $\mathcal{SPN}$, each place holds a discrete number of tokens to describe a specific state of the modelled object, e.g. a species. Each stochastic transition enjoys a stochastic firing rate function determining a stochastic waiting time before the actual firing takes place. Similarly, $\mathcal{SPN}^{\mathcal{C}}$ comprise discrete places and stochastic transitions. Additionally, each discrete place gets assigned a colour set permitting to specify a multi-set colour expression as place marking. The multi-set colour expression comprises a set of elements (colours) with their corresponding occurrences. Assuming the colour set CS ={a, b, c} (three colours), the multi-set expression $\{1`a{+}{+}3`b{+}{+}5`c\}$ describes that there is one occurrence of the colour a, three occurrences of the colour b, and five occurrences of the colour c. It is worth mentioning that multi-set expressions may be used to decorate arcs connecting coloured places and transitions for the purpose of determining the coloured tokens that will flow over arcs. Additional to the stochastic rate functions, each coloured stochastic transition gets assigned a guard, which is a Boolean expression (constraint) determining the colour bindings, for which the corresponding transition may fire. In the colour world, a transition gets enabled if the following condition holds.

$$\forall p \in {}^{\bullet}t, m(p) \geq f(p,t)\langle B\rangle, \tag{1}$$

where ${}^{\bullet}t$ denotes the pre-places of the transition t, $f(p,t)$ denotes the multi-set colour expression of the arc connecting the place p to the transition t, and $\langle B\rangle$ denotes a binding set. The binding set holds all colour combinations induced by the variables that are involved in the colour expressions of the transition's adjacent arcs, for which the transition's guard expression is evaluated to *True* [22]. The condition given in Equation 1 describes that the transition t is said to be enabled using each binding $b(t) \in B$, if all the pre-places of t hold a number of coloured tokens greater than or equal to the corresponding evaluated colours (over arcs). In the following, we give the formal definition of $\mathcal{SPN}^{\mathcal{C}}$.

Definition 1 (Coloured stochastic Petri net [12]):
A *coloured stochastic Petri net* is a 9-tuple
$N = \langle P, T, A, \Sigma, c, g, f, v, m_0\rangle$, where:

- P is a finite, non-empty set of places.
- T is a finite, non-empty set of transitions.
- $P \cap T = \emptyset$
- $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of directed arcs.
- Σ is a finite, non-empty set of colour sets.
- $c : P \to \Sigma$ is a colour function that assigns to each place $p \in P$ a colour set $c(p) \in \Sigma$.
- $g : T \to EXP$ is a guard function that assigns to each transition $t \in T$ a guard expression of the Boolean type.
- $f : A \to EXP$ is an arc function that assigns to each arc $a \in A$ an arc expression of a multiset type $c(p)_{MS}$, where p is the place adjacent to the arc a.
- v: is a function that assigns a stochastic rate function to each transition $t \in T$. This rate function determines a stochastic waiting time before the transition t may fire.
- $m_0 : P \to EXP$ is an initialisation function that assigns to each place $p \in P$ an initialisation expression of a multiset type $c(p)_{MS}$.

For illustrating this formal definition, we give an $\mathcal{SPN}^{\mathcal{C}}$ model of the scaleable repressilator to be used later as running example. The repressilator [23], [24] is a synthetic genetic network serving as a model to study gene regulatory networks and the dynamics of genetic oscillations. Notably, the repressilator is scaleable by the number of genes involved in the genetic network. Each gene produces a repressor protein that inhibits the expression of the next gene in a cyclic way. The produced proteins can also be degraded. Note that each reaction in the system occurs according to Mass-Action kinetics rate functions[a], see Table I.

Figure 1 gives a coloured stochastic Petri net model of the repressilator. First, the colour set *GeneSet* is defined, which is an *enum* colour set, whose elements represent the

TABLE I
RATE FUNCTIONS OF THE REPRESSILATOR SYSTEM'S REACTIONS

reaction	***Rate function***	***Kinetic parameter***
generate	$k\_gen \times gene$	k_gen
block	$k\_block \times protein \times gene$	k_block
degrade	$k\_deg \times protein$	k_deg
unblock	$k\_unblock \times blocked$	$k\_unblock$

[a]MassAction(k) = k $\prod power(p, w)$ with $p \in {}^\bullet t$

set of genes in the systems, i.e. *enum GeneSet* = $\{a, b, c\}$. Next, one variable of the type $GeneSet$, e.g. x is declared to encode a gene to be regulated in the network. Then, the model is constructed by adding three coloured places that get assigned the colour set *GeneSet*. Each place represents system states that are either *gene*, *protein* or *blocked*. The place *gene* is initialised with one token of each colour to describe that the system starts with three genes with one instance each. This is reflected in the model by assigning the colour function $1`all()$ to the place *gene*, determining one token of each colour yielding three coloured tokens. Moreover, all reactions that share the copies of reactants/products, e.g. genes/proteins are represented using one coloured stochastic transition whose rate function gets assigned according to Table I. For instance, the coloured transition *generate* is a high-level transition representing three reactions, as the colour set *GeneSet* is assigned to its pre-place and post-place and this colour set comprises three colours. It is worth mentioning that in this example net transitions do not get guard expressions, which means that there are no constraints on enabling the net transitions. Afterwards, standard arcs (directed arcs) decorated with colour expressions are added connecting transitions and places to express the required net structure of the system.

The crucial advantage of the colouring is that the size of the modelled system can easily be adjusted by changing the colour sets of the model. For example, changing the colour set *GeneSet* by, e.g., adding more colours will scale the circular gene network according to the newly added colours without having to touch the net structure, i.e. the net places/transitions and their connectivity style.

II. Approach

A. Coloured stochastic Petri nets semantics

In order to present our high-level simulation algorithm, we are going to recall the basic principles of simulating standard stochastic Petri nets. As uncoloured stochastic Petri nets, the underlying semantic of $\mathcal{SPN}^\mathcal{C}$ is defined by continuous-time Markov chains (CTMC) [26]. However, deriving the complete CTMC from $\mathcal{SPN}$ may be infeasible, as the state space induced by the corresponding reachability graph can be infinite. Therefore, the CTMC is approximated by simulating an $\mathcal{SPN}$ model. Building the approximated CTMC can be achieved by generating sufficiently many different paths through it. To this end, starting from the initial marking, the net transitions have to be fired repeatedly [7]. To accomplish this, two questions arise, which have to be answered. These questions are the

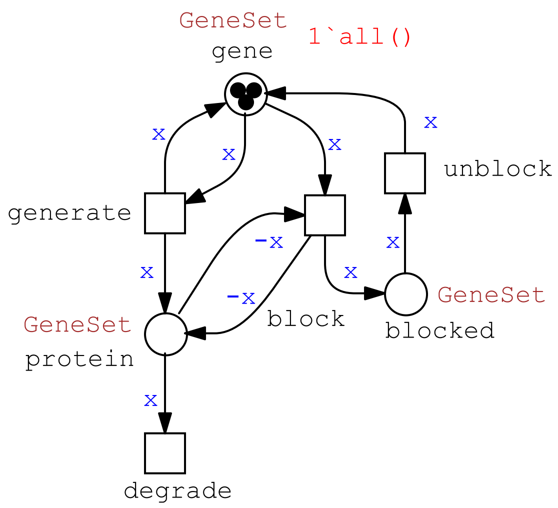


Fig. 1. $\mathcal{SPN}^\mathcal{C}$ of the scalable repressilator system, which is modelled using *Snoopy* [25]. The model can be scaled by adding new colours, i.e. gene identifiers to the *enum* colour set *GeneSet*. Note that the colour set *GeneSet* could also be defined as *integer* type. In this model, rate functions of the coloured transitions are not shown for the sake of readability.

following. First, when will the next transition fire? Second, which transition will fire? Gillespie's stochastic simulation algorithm [27] (known as Gillespie's SSA) is adopted to answer these two questions.

According to Gillespie's SSA, each reaction, i.e. a $\mathcal{PN}$ transition r_j defines a propensity function (also known as hazard function) $p_j(r)$ with the parameter k_j representing *mass-action* semantic, compare Equation 2

$$p_j(r) = k_j \prod_{i=1; p_i \in {}^\bullet t}^{N_j} \binom{m_{(p_i)}}{a_{ji}} \tag{2}$$

Here, m_i is the number of molecules of one reactant (a pre-place p_i) of r_j, and a_{ji} gives the corresponding stoichiometry. N_j is the number of reactants (pre-places) of the reaction (transition) r_j. Then, the total propensity is calculated over all reactions using Equation 3.

$$TP_0(r) = \sum_{j=1}^{M} p_j(r) \ , \tag{3}$$

with M being the total number of the system's reactions, i.e. net transitions. In order to determine the time step at which the next reaction will occur as well as to choose the reaction to occur, the algorithm has to generate two random numbers r_1 and r_2 according to the uniform distribution on the interval (0,1). The next time step is calculated using Equation 4.

$$N_t = -\frac{1}{TP_0(r)} ln r_1 \tag{4}$$

Afterwards, the reaction to occur R_μ is chosen according to Equation 5.

$$\sum_{j=1}^{\mu-1} p_j(r) < r_2 TP_0(r) \leq \sum_{j=1}^{\mu} p_j(r) \tag{5}$$

The detailed algorithmic implementation of the $\mathcal{SPN}$ simulation based on Gillespie's SSA can be found in [7].

The former steps are directly performed on uncoloured $\mathcal{SPN}$ as the system reactions and the relevant species are explicitly given by mapping them to stochastic transitions and discrete places. However, simulating biological systems modelled as $\mathcal{SPN}^{\mathcal{C}}$ is not trivial, as the give $\mathcal{SPN}^{\mathcal{C}}$ model has to be unfolded into the corresponding uncoloured $\mathcal{SPN}$ counterpart before the actual stochastic simulation is performed, using the unfolded net in the background. $\mathcal{SPN}^{\mathcal{C}}$ unfolding includes unfolding all the coloured places, transitions and connecting arcs. In this context, net unfolding is time- as well as space-consuming as it requires solving the constraint satisfaction problem (CSP) induced by transition guards. For example, unfolding our running example given in Figure 1 produces the $\mathcal{SPN}$ presented in Figure 2. More details on unfolding of coloured Petri nets can be found in [15].

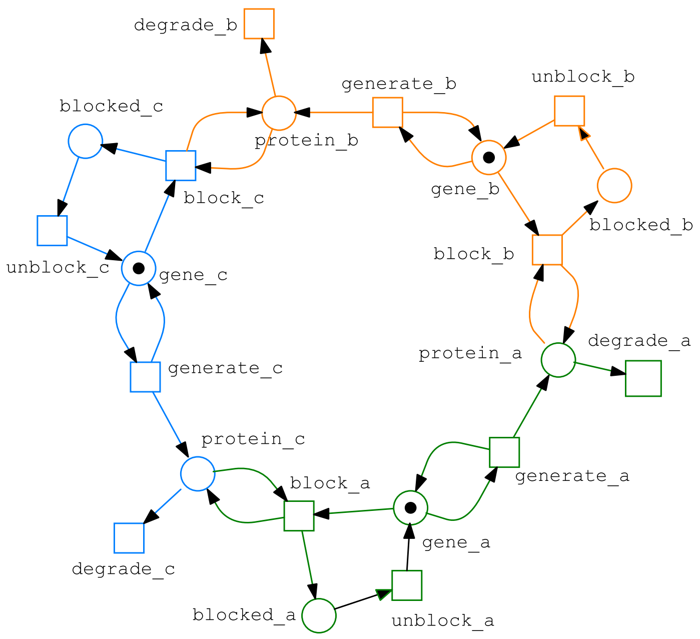


Fig. 2. Unfolded $\mathcal{SPN}$ model of the repressilator $\mathcal{SPN}^{\mathcal{C}}$ shown in Fig.1 comprising 9 discrete places, 12 stochastic transitions, and 36 arcs. Note that net elements highlighted for illustration with the same colour belong to the same gene activities (component). This implies that the three gene activities have the same net structure.

In the following, we introduce our new approach for simulating $\mathcal{SPN}^{\mathcal{C}}$ models on-the-fly without having to unfold the coloured net as a preliminary step.

B. High-level stochastic simulation algorithm

Our new solution for simulating $\mathcal{SPN}^{\mathcal{C}}$ models without the unfolding step is basically tuning Gillespie's SSA algorithm (see former Section) to work directly on the colour level. The proposed algorithm has to perform the following steps.

1) Simulator initialisation: includes writing the initial state of the $\mathcal{SPN}^{\mathcal{C}}$ model, i.e. the initial marking of the coloured places at time $\tau = 0$ to the output trace file. In pursuit of this, the algorithm has to iterate over all the coloured places and then write the number of tokens of each colour (of every place) to the output trace file. Furthermore, the output trace file is initialised with the colour marking of every coloured place, i.e. the sum over all coloured tokens of every coloured place. It is worth mentioning that the evolution of a coloured place over time is obtained by obtaining the summation of its coloured tokens as the simulation evolves. This measure is an interesting measure for various biological systems; for instance, the total number of molecules over time of all genes with a specific type . However, this measure is not considered by the standard stochastic simulation algorithm. Algorithm 1 sketches these steps.

Note that the algorithm determines the identifiers of the output variables (places) by concatenating the place identifier with the colour identifiers of the place's colour set (line 6). Each obtained output variable is called a place instance. Crucially, the algorithm will only initialise output traces for the set of place instances given as input to the initialisation algorithm, so that only the variables of interest will be written to the output instead of writing all place instances.

2) Determining the time at which the next reaction will occur: As in the standard Gillespie algorithm, the high-level simulation algorithm has to determine the time step at which the next transition will occur. For this purpose, we introduce the colour propensity function, see Equation 6.

$$p_j{}^b(r) = k_j \prod_{i=1; p_i \in {}^\bullet t_b}^{N_j} \binom{m_{(p_i)}}{a_{ji}{}^b} \tag{6}$$

Algorithm 1 Initialise output traces

Input: $\mathcal{SPN}^{\mathcal{C}}$ model with its initial state $m(\tau_0)$, S is the set of selected output variables.
Output: Output traces of selected place instances/coloured places at time τ_0.

```
1:  for each coloured place $p \in P$ do
2:      if $p \in S$ then
3:          write overall marking of $p$;
4:      end if
5:      for each colour $c \in C(p)$ do
6:          write place instance id such that concat(placeId(p), "_", colourId(c));
7:          if $placeinstanceId \in S$ then
8:              if colour $c \in m(p)$ then
9:                  write the occurrence of $c$ as initial state;
10:             else
11:                 write 0 as initial marking;
12:             end if
13:         end if
14:     end for
15: end for
```

Where
${p_j}^b(r)$ is the propensity function corresponding to one transition (reaction) whose binding is b (a transition instance),
${}^\bullet t_b$ gives the pre-places of the corresponding transition with the binding b,
$m_{(p_i)}$ gives the occurrence of the colour binding b in the coloured pre-place p_i of t,
${a_{ji}}^b$ gives the occurrence of colour binding b over the arcs connecting the coloured pre-place p_i of t.

Then, the total coloured propensity is obtained using Equation 7.

$$TCP_0(r) = \sum_{j=1}^{M} {p_j}^b(r) \tag{7}$$

For determining the next firing time, the algorithm has to follow the same principle as the standard Gillespie's SSA by generating two random numbers r_1 and r_2 according to the uniform distribution. Then, the next firing time is obtained using Equation 8.

$$N_{tc} = -\frac{1}{TCP_0(r)} ln r_1 \tag{8}$$

3) Determining the next reaction to occur: Choosing a reaction (a colour transition) to occur is equivalent to choosing a colour transition t whose index is μ, together with one binding b, for which the transition t is enabled. To obtain these two pieces of information, Equation 5 needs to be adjusted as given in Equation 9.

$$\sum_{j=1}^{\mu-1} {p_j}^b(r) < r_2 TCP_0(r) \leq \sum_{j=1}^{\mu} {p_j}^b(r) \tag{9}$$

4) Removing isolated places: Isolated places [15] are the places that are not directly reachable from the initial marking of a Petri net. Such places may be generated as there is no variable binding enabling a certain coloured transition due to, e.g., a never fulfilled guard. This means that these places do not contribute to the model behaviour, i.e their marking never changes over time. Therefore, these places can be safely excluded from the output trace file. Algorithm 2 cleans the output traces from the isolated places, if they exist.

Algorithm 2 Removing isolated places.

```
Input: $\mathcal{SPN}^{\mathcal{C}}$ output traces.
Output: Output traces over time excluding isolated places.
1: for each instance place do
2:     if no evolution occurs over time then
3:         get rid of the corresponding column;
4:     end if
5: end for
```

Algorithm 3 sketches the steps required for simulating $\mathcal{SPN}^{\mathcal{C}}$ models. The algorithm takes an $\mathcal{SPN}^{\mathcal{C}}$ model together with the desired simulation end time and produces model states over time as output. First, the simulator initialisation is performed (lines 1-2). This includes setting simulation time ($\tau = 0$) and writing the initial system state of the model to the output trace file using Algorithm 1. Then, the simulator loop (lines 3 -18) has to be performed until the simulator time reaches the end time (τ_{end}). Afterwards, the simulation loop is started by iterating over coloured transitions. For each enabled coloured transition, the algorithm computes both colour propensity function as well as the total colour propensity using Equations 6 and 7 (lines 7-8), respectively. After that, two random numbers are generated according to the uniform distribution (line 12). Then, the time step at which the next transition will occur is obtained by applying Equation 8, and the transition to occur (transition firing) together with the corresponding binding is selected by applying Equation 9 (lines 13-14). At this point, the chosen coloured transition t is fired using the binding b (line 15). As a result, the simulator clock is advanced by the determined next time step (line 16) and the system state together with output traces are updated accordingly (line 17). Finally, isolated places are removed by performing Algorithm 2 (line 19).

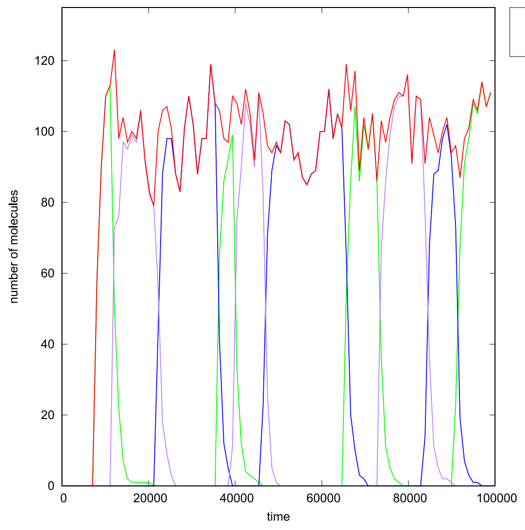
It is worth mentioning that simulation traces can be exactly reproduced by seeding the utilised random number generator with the very same value, whenever the simulation algorithm is performed on a given model [28]. This feature permits to conveniently verify the correctness of the new algorithm and its implementation by comparing its output traces with those generated by simulating the corresponding unfolded model using the standard Gillespie's SSA, as the standard simulation algorithm implementation in Snoopy allows setting an option to seed the random number generators.

Figure 3 presents simulation traces of the scaleable repressilator $\mathcal{SPN}^{\mathcal{C}}$ model given in Figure 1 by performing the high-level stochastic simulation algorithm. Sub-figure (a) gives the traces of the produced proteins together with the corresponding total protein evolution (coloured place evolution). Sub-figure (b) shows the traces of the produced proteins for the $\mathcal{SPN}^{\mathcal{C}}$ model with 7 genes ranging from *a* to *g*. For both cases, the oscillation behaviour of the systems is obviously captured.

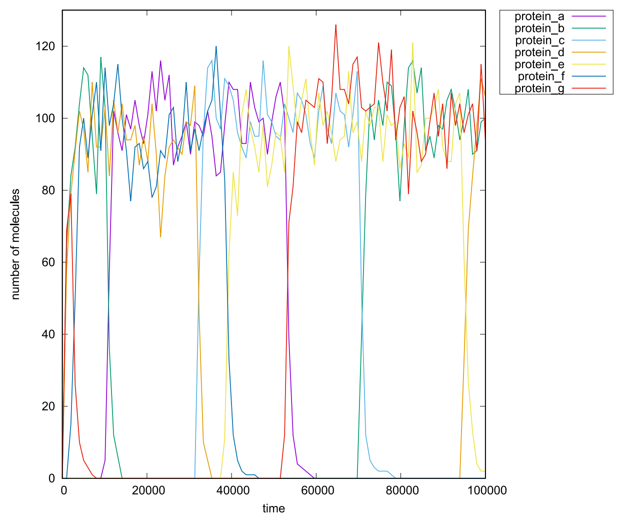
III. ANOTHER CASE STUDY

In this section, we present another biological system, known as the *circadian rhythm* [29]. Circadian rhythm is an internal timing system that regulates various physiological and behavioural processes, e.g. the sleep/wake cycle in living organisms. The system comprises a set of genes (clock genes) that regulate their own transcription and translation in a cycle that takes approximately 24 h to complete [29], [30]. Assuming a system of two clock genes, e.g. *A, R*, gene transcription yields an *mRNA* whose translation produces a protein. While the first gene positively regulates its transcription, the second gene negatively regulates the corresponding transcription.

Figure 4 presents an $\mathcal{SPN}^{\mathcal{C}}$ model of the *circadian rhythms*. To encode two types of clock genes, we declare a colour set of *enum* type with two colours, i.e. *enum Genes* = {*A, R*} . Further, one variable of the type *Genes* is declared, i.e.,



(a)



(b)

Fig. 3. Stochastic simulation traces of the repressilator $\mathcal{SPN}^{\mathcal{C}}$ model. For reproducibility purposes, random numbers have been seeded with the (arbitrarily chosen) value 1895243956. (a) Scaleable $\mathcal{SPN}^{\mathcal{C}}$ model with 3 genes. (b) Scaleable $\mathcal{SPN}^{\mathcal{C}}$ model with 7 genes, namely *a,b,c,d,e,f,g*.

Algorithm 3 High-level stochastic simulation

Input: $\mathcal{SPN}^{\mathcal{C}}$ model with its initial state $m(\tau_0)$, simulation end time τ_{end}, *seed value*.

Output: Output traces of model variables (places).

```
1:  initialise simulator clock $\tau$ = 0.0;
2:  initialise output traces using Algorithm 1;
3:  while $\tau < \tau_{end}$ do
4:      for each coloured transition $t \in T$ do
5:          if $t$ is enable then
6:              for each binding $b \in B$ do
7:                  obtain colour propensity using Equation 6;
8:                  compute total propensity using Equation 7;
9:              end for
10:         end if
11:     end for
12:     generate two random variables $r_1$ and $r_2$ using $seed$;
13:     compute time step $N_{tc}$ using Equation 8;
14:     select coloured transition with one binding using Equation 9;
15:     fire colour transition $t$ using binding $b$;
16:     advance simulator clock $\tau = \tau + N_{tc}$;
17:     update system state together with output traces;
18: end while
19: remove isolated places using Algorithm 2;
```

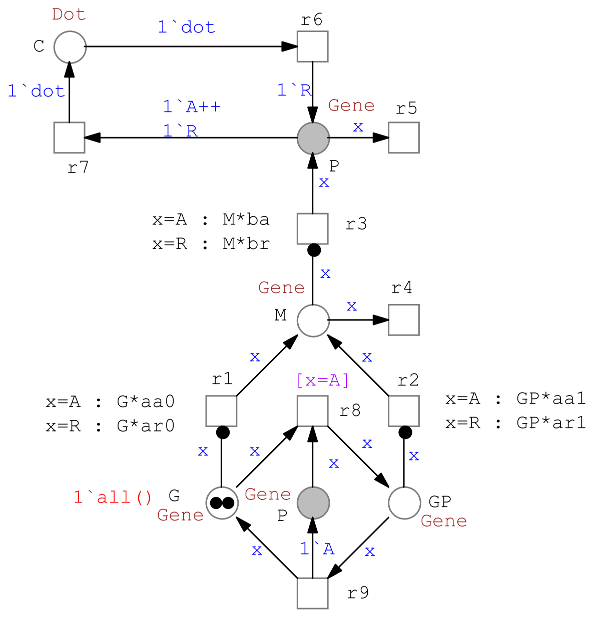
x *: Genes*. Then, the coloured net is obtained by folding (colouring) the basic $\mathcal{SPN}$ model introduced in [10], [12]. The transcription of each gene (modelled by place G) is expressed by the transition r_1 which consequently produces an $mRNA$ molecule (modelled by place M), which in turn either degrades (r_4) or translates into a protein (place P). To learn more about modelling this sequence of biological activities together with the required modelling means, we recommend to consult [31].

TABLE II
RATE FUNCTIONS OF THE CIRCADIAN RHYTHM. THE COLUMN *Gene* REPRESENTS THE COLOUR DEPENDENCIES OF RATE FUNCTIONS.

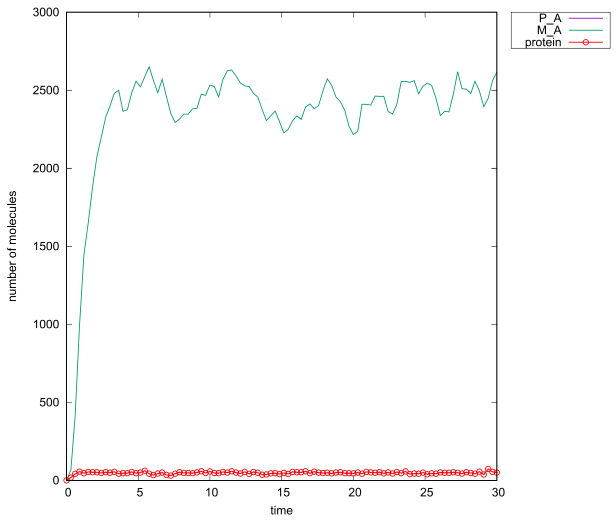
reaction	***Gene***	***Rate function***	***Kinetic parameter***
r1	A	$G \times aa_0$	aa_0
	R	$G \times ar_0$	ar_0
r2	A	$GP \times aa_1$	aa_1
	R	$GP \times ar_1$	ar_1
r3	A	$M \times ba$	ba
	R	$M \times br$	br
r4	A	$M \times dam$	dam
	R	$M \times drm$	drm
r5	A	$P \times da$	da
	R	$P \times dr$	dr
r6	A, R	$C \times da$	da
r7	A, R	$P \times gar$	gar
r8	A	$P \times G \times ga$	ga
	R	$P \times G \times gr$	gr
r9	A	$GP \times ta$	ta
	R	$GP \times tr$	tr

In contrast to the running repressilator example, reaction rates of the *circadian rhythm* are colour-dependent, which means that, e.g., the transcription of one gene occurs with a different rate constant than the transcription of another gene. For instance, the reaction r_1 has colour-dependent rates, describing that the transcription of the gene A occurs with rate constant $aa0$, whereas the transcription of gene R occurs with rate constant $ar0$, compare Table II for the rate functions of the model transitions. Note that not all rate functions are shown in the $\mathcal{SPN}^{\mathcal{C}}$ model given in Figure 4 in order to obtain a better layout.

Figure 4 (Sub-figure b) gives stochastic simulation traces of both mRNA and the produced protein of the gene A, namely M_A and p_A, respectively.



(a)



(b)

Fig. 4. (a) Coloured stochastic Petri net of the circadian rhythm system, in which the system is initialised with one molecule of each gene. Note that not all rate functions are shown in the model to enhance readability. (b) Stochastic simulation traces of the *mRNA* and protein produced by gene *A*. For reproducibility purposes, random numbers have been seeded with the (arbitrarily chosen) value 1624402775.

IV. Discussion

The newly proposed algorithm has shown that exploring behaviours of biological systems with similar structures can be achieved on the fly without having to flatten the modelled system into its basic reactions. The major advantages that are offered by our algorithm are as follows. First, there is no need to unfold coloured places, transitions and arcs to their uncoloured counterparts, which saving memory space. In this context, our algorithm offers to choose the variables of interest, i.e. places to be recorded in the simulation trace file. Second, our algorithm introduces a measure of recording the evolution of high-level variables, i.e. coloured places over time which is obtained on-the-fly rather than post-calculating, compared to the traditional stochastic simulation.

For the purpose of evaluating the performance of our algorithm, we run both the standard stochastic simulation algorithm and the high-level simulation algorithm, both implemented in *Snoopy* [25], on the presented running example (the repressilator). Table III summarises the facts representing the time spent by both algorithms against different settings of simulation. Note that the random number generator has been seeded with a random seed number per each experiment^b.

Table III shows that the standard Gillespie's stochastic simulation algorithm outperforms the current implementation of the high-level stochastic simulation algorithm by its run time. One reason for this can be seen in the following.

In both standard simulation and high-level simulation, most of the run time is spent for calculating the next time point at which the next transition will fire. This piece of information is obtained by calculating the propensity functions of all enabled (coloured) reactions, i.e. enabled transitions, for every iteration of the simulator loop. In this context, firing one transition per iteration will influence only those transitions that directly depend on the pre/post-places of the fired one. Notably, *Snoopy's* implementation of the standard simulation algorithm exploits this fact by only updating the propensity functions of those transitions that have been affected by the previous transition's firing [7]. The current implementation of the high-level simulation algorithm does not consider this fact so far, but it will be subject for further work (see next Section).

TABLE III
Performance evaluation

End time (time unit)	High-level simulation	Standard simulation
50	1.24 sec	$<$ 1 sec
100	3.73 sec	$<$ 1 sec
500	6.75 sec	$<$ 1 sec
1000	21.93 sec	$<$ 1 sec
2000	38.95 sec	$<$ 1 sec
10 000	3 m 21 sec	$<$ 1 sec
100 000	26 m 7 sec	1.01 sec

^{b}Experiments have been performed on Windows machine with Core i7 CPU/1.80 GHz, RAM capacity of 32 GB.

V. Conclusions

In this paper, we introduced a new stochastic simulation algorithm for analysing the behaviour of biological systems characterised by having components with similar structures. Our proposed algorithm is based on adapting the well-known Gillespie's stochastic simulation algorithm to work directly on the colour level, which comes with the prodigious advantage of saving memory space.

Furthermore, we presented two coloured stochastic Petri net models for two biological case studies. Performing the high-level stochastic simulation algorithm reproduced results identical to the ones of the standard stochastic simulation algorithm. For reproducibility purposes, our presented models can be accessed via [32]. The latest version of *Snoopy* [25] is accessible via [33]. Snoopy has a couple of features to plot the simulation traces of the variables of interest as well as to export the selected traces to, e.g. *.csv* format to be plotted using external tools. To guide yourself on using *Snoopy* for modelling and simulating $\mathcal{SPN}^{\mathcal{C}}$ models, please consult our technical report [32].

The current implementation of the high-level stochastic simulation algorithm considers so far stochastic transitions only. The upcoming version of our algorithm will consider some extensions too, e.g., generalised coloured stochastic Petri nets which provide other transition types than stochastic transitions, such as *immediate transitions* and *scheduled transitions* together with a set of special arcs, e.g., *inhibitory arcs*.

Moreover, future work will include improving the efficiency of our algorithm by constructing the dependency graph of the coloured net as discussed in Section IV, which is expected to substantially boost its runtime performance.

Further work will also improve on the user support by allowing to write a pattern formula for the transitions' rate functions, e.g. *MassAction(k)* instead of writing the pre-places explicitly.

Last but not least, for some biological systems, it is interesting to gain knowledge on the number of event occurrences, i.e. transition firings and their schedule as the simulator evolves. This will also be included in the course of our future work package.

Acknowledgment

We would like to thank David Gilbert for many fruitful discussions.

References

[1] O. Pârvu, D. Gilbert, M. Heiner, F. Liu, N. Saunders, and S. Shaw, "Spatial-temporal modelling and analysis of bacterial colonies with phase variable genes," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 25, no. 2, p. 25p., May 2015.

[2] F. Liu, M. Heiner, and D. Gilbert, "Coloured Petri nets for multilevel, multiscale, and multidimensional modelling of biological systems," *Briefings in Bioinformatics*, vol. 20, no. 3, pp. 877–886, 2019.

[3] A. Ismail, M. Herajy, E. Atlam, and M. Heiner, "A Graphical Approach for Hybrid Simulation of 3D Diffusion Bio-models via Coloured Hybrid Petri Nets," *Modelling and Simulation in Engineering*, vol. 2020, 2020.

[4] S. Connolly, D. Gilbert, and M. Heiner, "From Epidemic to Pandemic Modelling," *Frontiers in Systems Biology*, March 2022.

[5] W. Reisig, "Petri nets and algebraic specifications," *Theoretical Computer Science*, vol. 80, no. 1, pp. 1–34, 1991.

[6] M. Herajy, "Computational steering of multi-scale biochemical networks," Ph.D. dissertation, BTU Cottbus, Dep. of CS, January 2013.

[7] C. Rohr, "Simulative analysis of coloured extended stochastic petri nets," Ph.D. dissertation, BTU Cottbus, Dep. of CS, January 2017.

[8] M. Herajy and M. Heiner, "Adaptive and Bio-semantics of Continuous Petri Nets: Choosing the Appropriate Interpretation," *Fundamenta Informaticae*, vol. 160, no. 1-2, pp. 53–80, 2018.

[9] M. Blätke, C. Rohr, M. Heiner, and W. Marwan, *A Petri Net based Framework for Biomodel Engineering*, ser. Modeling and Simulation in Science, Engineering and Technology. Springer, Birkhäuser Mathematics, December 2014, pp. 317–366.

[10] M. Blätke, M. Heiner, and W. Marwan, *BioModel Engineering with Petri Nets*. Elsevier Inc., March 2015, ch. 7, pp. 141–193.

[11] K. Jensen, "Coloured petri nets," in *Petri Nets: Central Models and Their Properties*, W. Brauer, W. Reisig, and G. Rozenberg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987.

[12] F. Liu, "Colored petri nets for systems biology," Ph.D. dissertation, BTU Cottbus, Dep. of CS, 2012.

[13] G. Assaf, " Fuzzy coloured Petri nets for modelling biological systems with uncertain kinetic parameters ," Ph.D. dissertation, BTU Cottbus, Dep. of CS, 2022.

[14] M. H. A Ismail and M. Heiner, *A Graphical Approach for the Hybrid Modelling of Intracellular Calcium Dynamics Based on Coloured Hybrid Petri Nets*, ser. Computational Biology. Springer, Cham, 2019, vol. 30, ch. 13, pp. 349–367.

[15] M. Schwarick, C. Rohr, F. Liu, G. Assaf, J. Chodak, and M. Heiner, "Efficient unfolding of coloured petri nets using interval decision diagrams," in *Proc. PETRI NETS 2020*, ser. LNCS, T. C. R Janicki, N Sidorova, Ed., vol. 12152. Springer, Cham, June 2020, pp. 324–344.

[16] Q. Gao, D. Gilbert, M. Heiner, F. Liu, D. Maccagnola, and D. Tree, "Multiscale Modelling and Analysis of Planar Cell Polarity in the Drosophila Wing," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 2, pp. 337–351, 2013.

[17] F. Liu, M. Blätke, M. Heiner, and M. Yang, "Modelling and simulating reaction–diffusion systems using coloured petri nets," *Computers in Biology and Medicine*, vol. 53, pp. 297–308, 2014.

[18] F. Liu and M. Heiner, "Modeling membrane systems using colored stochastic petri nets," *Nat. Computing*, vol. 12, no. 4, pp. 617 – 629, 2013.

[19] ——, "Multiscale modelling of coupled ca2+ channels using coloured stochastic petri nets," *IET Systems Biology*, vol. 7, no. 4, pp. 106 – 113, August 2013.

[20] M. Heiner, S. Lehrack, D. Gilbert, and W. Marwan, "Extended stochastic petri nets for model-based design of wetlab experiments," *Transactions on Computational Systems Biology XI*, vol. 5750, pp. 138–163, 2009.

[21] C. Rohr, "Discrete-Time Leap Method For Stochastic Simulation," *Fundamenta Informaticae*, vol. 160, no. 1-2, pp. 181–198, 2018.

[22] L. M. Kristensen and S. Christensen, "Implementing coloured petri nets using a functional programming language," *Higher-Order and Symbolic Computation*, vol. 17, no. 3, pp. 207–243, Sep 2004.

[23] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, Jan 2000.

[24] F. Liu and M. Heiner, *Petri Nets for Modeling and Analyzing Biochemical Reaction Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 245–272.

[25] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick, "Snoopy – a unifying petri net tool," in *Application and Theory of Petri Nets*, S. Haddad and L. Pomello, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 398–407.

[26] M. Heiner, D. Gilbert, and R. Donaldson, "Petri nets for systems and synthetic biology," in *Formal Methods for Computational Systems Biology*, M. Bernardo, P. Degano, and G. Zavattaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 215–264.

[27] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976.

[28] J. Chodak, "Spike – a tool for reproducible simulation experiments," Ph.D. dissertation, BTU Cottbus, Dep. of CS, December 2022.

[29] D. Gonze, J. Halloy, and A. Goldbeter, "Deterministic versus stochastic models for circadian rhythms," *Journal of Biological Physics*, vol. 28, pp. 637–653, Dec 2002.

[30] S. Hood and S. Amir, "Biological clocks and rhythms of anger and aggression," *Frontiers in Behavioral Neuroscience*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbeh.2018.00004

[31] F. Liu, G. Assaf, M. Chen, and M. Heiner, "A Petri nets-based framework for whole-cell modeling," *BioSystems*, vol. 210, p. 104533, 2021.

[32] (2023) Coloured stochastic petri net models. [Online]. Available: https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Examples

[33] (2023) Snoopy - petri nets modelling and simulation tool. [Online]. Available: https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy