Formalisms for Specifying Markovian Population Models

Thomas Henzinger, Barbara Jobstmann, and Verena Wolf

School of Computer & Communication Sciences, EPFL, 1015 Lausanne, Switzerland

Abstract. We compare several languages for specifying Markovian population models such as queuing networks and chemical reaction networks. These languages —matrix descriptions, stochastic Petri nets, stochastic process algebras, stoichiometric equations, and guarded command models— all describe continuous-time Markov chains, but they differ according to important properties, such as compositionality, expressiveness and succinctness, executability, ease of use, and the support they provide for checking the well-formedness of a model and for analyzing a model.

1 Introduction

Markov chains are an omnipresent modeling approach in the applied sciences. Often, they describe *population processes*, that is, they operate on an multi-dimensional discrete state space, where the dimensions correspond to certain types of individuals. Depending on the application area, "individuals" are customers in a queuing network, molecules in a chemically reacting volume, servers in a computer network, etc.

Here, we are particularly interested in dynamical models of biochemical reaction networks, such as signaling pathways, gene expression networks, and metabolic networks. They are an important emerging application area of continuous-time Markov chains and operate on an abstraction level where a state of the system is given by an n-dimensional vector of chemical populations, that is, the systems involves n different types of molecules and the i-th entry is the number of molecules of type i. Molecules collide randomly and may undergo certain chemical reactions, which change the state of the system. Classical modeling approaches in biochemistry are based on a system of ordinary differential equations that assume a continuous deterministic change of chemical concentrations. During the last decade, however, various experimental results have shown that the

discreteness and randomness of the chemical reactions have to be taken into account. Thus, discrete-state Markov models have gained in importance to describe the dynamics in the cell [32, 37, 41, 34, 44, 46, 47].

Many different formalisms for the specification of Markovian population models exist. Most popular are matrix descriptions, stochastic Petri nets, stochastic process algebras, and languages based on guarded commands. Moreover, Markov chains for biochemical reaction networks are often specified based on rules for chemical reactions, called stoichiometric equations.

In this paper we give a brief survey on modeling formalisms for Markovian population models and discuss important syntax-related aspects, such as compositionality, expressiveness and succinctness, executability, and well-formedness. We illustrate our considerations with examples of population models. We highlight that, even though all specification languages that we consider describe the same models, the choice of the language has significant implications for the modeling process itself, but also for the analysis of the model.

2 Continuous-Time Markov Chains

Let S be a countable set. We consider a (homogeneous) continuous-time Markov chain $(X^{(t)})_{t\geq 0}$ on a probability space $(\Omega, \mathcal{F}, Pr)$ with state space S and transition function

$$P_{ij}^{(t)} = Pr(X^{(t)} = j \mid X^{(0)} = i), \quad i, j \in S, t \geq 0.$$

If initial probabilities $Pr(X^{(0)} = i)$ are specified for each $i \in S$, the transient state probabilities $p_j^{(t)} := Pr(X^{(t)} = j)$, are given by

$$p_j^{(t)} = \sum\nolimits_{i \in S} p_i^{(0)} \cdot P_{ij}^{(t)}.$$

The transition functions $P_{ij}^{(t)}$ of a Markov chain are usually represented by their derivatives $q_{ij} = P'_{ij}(0)$ at $t = 0$, called *rates*. Here, we focus on Markov chains arising from population models. We therefore rule out "pathological cases" by assuming that the rates are finite and that $\sum_{j \in S} q_{ij} = 0$ for all $i \in S$. Note that this ensure that the sample paths $X^{(t)}(\omega)$ are right-continuous step functions

(at least until a certain random point in time). Let Q be the matrix with components q_{ij}. Note that the diagonal elements are nonpositive and the off-diagonal elements are nonnegative. The matrix Q is called the (infinitesimal) *generator* of X since, if $\sup_{i \in S} |q_{ii}| < \infty$, the transition functions can be "generated" from Q. They are the unique solution of the Kolmogorov backward equations

$$P'(t) = Q \cdot P(t), \tag{1}$$

where the components of $P(t)$ are the values $P_{ij}(t)$. As a general solution, this gives

$$P(t) = \exp(Qt) = \sum\nolimits_{k=0}^{\infty} (Qt)^k / k!.$$

Algorithms for the computation of the vector $p^{(t)}$ with entries $p_j^{(t)}$ are usually based on the numerical integration of the linear system of differential equations

$$p'(t) = Q \cdot p(t), \tag{2}$$

with initial condition $p^{(0)}$. Another approach is the approximation of the matrix exponential $\exp(Qt)$, which gives an approximation of $p(t) = p^{(0)} \cdot P(t) = p(0) \cdot \exp(Qt)$. In the case of an infinite or very large state space, the computation of $p(t)$ is computationally very expensive or even infeasible. Accurate approximations are, however, possible if the model is truncated appropriately.

For every $i \in S$, the limit probability $\pi_i = \lim_{t \to \infty} p_i(t)$ exists, but π_i may be zero for all states $i \in S$. Under certain conditions, however, $\sum_{i \in S} \pi_i = 1$ and the vector π with entries π_i is computed as the unique solution of the linear equation system

$$0 = \pi \cdot Q, \quad \sum\nolimits_{i \in S} \pi_i = 1. \tag{3}$$

The distribution π is then called *steady-state distribution* or *stationary distribution*. Note that in this case $\pi_i > 0$ for all $i \in S$.

Each Markov chain has an associated state-transition graph, called *intensity graph*. It is a directed graph whose node set corresponds to the state space of the chain. It has an edge from state i to state j labeled by q_{ij} whenever $q_{ij} > 0$. The Markov chain is uniquely determined by its intensity graph.

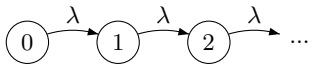

Fig. 1. The intensity graph of a Poisson process with rate λ.

Example 1. Consider a Markov chain X that has the infinitesimal generator

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & \dots \\ 0 & -\lambda & \lambda & 0 & \dots \\ \vdots & \ddots & \ddots & \ddots & \vdots \end{pmatrix},$$

where $\lambda > 0$. The process X is called Poisson process and is often used to model the number of arrivals of identical entities during a time interval $[0, t)$, where λ/t is the average number of arrivals per time unit. The intensity graph is shown in Fig. 1. □

3 Specifying Continuous-Time Markov Chains

In this section, we focus on the syntax of specification formalisms for large (or infinite) Markov chains with continuous time that describe *population models*, that is, models with state space $S = \mathbb{Z}_+^n = \{0, 1, \dots\}^n$, where the i-th state variable represents the number of instances of the i-th species. Depending on the application area, "species" stands for types of system components, molecules, customers, etc., and the application areas that we have in mind are chemical reaction networks, performance evaluation of computer systems, logistics, epidemics, etc.

3.1 Matrix Descriptions

A Markov chain may be specified by defining the elements of its generator matrix Q.

Example 2. Consider an epidemic process where individuals of a population are infected by a certain communicable disease. A state of the system is a pair $(x, y) \in \mathbb{Z}_+^2$, where x is the number of infected individuals and y is the number of individuals that are not

infected [38]. Given positive rate constants a, b, c, d, e, the positive elements of the (infinite) generator matrix Q are given by

$$\begin{array}{ll}
q_{(x,y),(x+1,y)} = a & \text{for } x \geq 0 \text{ and } y \geq 0, \\
q_{(x,y),(x-1,y)} = b \cdot x & \text{for } x > 0 \text{ and } y \geq 0, \\
q_{(x,y),(x,y+1)} = c & \text{for } x \geq 0 \text{ and } y \geq 0, \\
q_{(x,y),(x,y-1)} = d \cdot y & \text{for } x \geq 0 \text{ and } y > 0, \\
q_{(x,y),(x-1,y+1)} = e \cdot x \cdot y & \text{for } x > 0 \text{ and } y > 0.
\end{array}$$

All remaining off-diagonal entries are 0 and for each row the element on the diagonal is the negative sum of the remaining row entries. □

If the matrix exhibits a particular structure, it can also be described as the Kronecker product of smaller matrices that describe parts of the system. A general framework for descriptions based on the Kronecker product is provided by *stochastic automata networks* (SANs) [35, 15]. A stochastic automaton is equivalent to a state-transition graph in which directed edges are labeled by rates. Several automata can interact with each other and the state-transition graph of the global automaton determines the intensity graph of a Markov chain. The generator matrix of the Markov chain is then the Kronecker product of the matrices that represent the different automata and their interactions (compare also Section 4.1).

3.2 Stochastic Petri Nets

Petri nets are a pictorial language for describing systems of concurrent activities. A classical Petri net is a labeled directed bipartite graph whose node set is the disjoint union of a set P of *places* and a set T of *transitions*. The directed edges, called *arcs*, are given by a set $A \subseteq (P \times T) \cup (T \times P)$ and are labeled with a *multiplicity function* $l : A \to \mathbb{N}$. Stochastic Petri nets (SPN) [21] are an extension of classical Petri nets that associate a firing rate λ_τ with each transition $\tau \in T$.

A Petri net represents an infinite state-transition system with a set of states called *markings*. A marking is a function $m : P \to \mathbb{N}$ that maps every place of the Petri net to a nonnegative integer representing the number of *tokens* in that place. Given a marking m, a transition $\tau \in T$ is *enabled in* m if all places p with an arc a leading to τ have at least $l(a)$ tokens in m, i.e., $m(p) \geq l(a)$. Note that

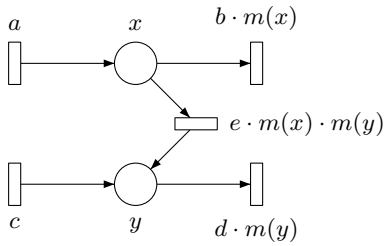

Fig. 2. Stochastic Petri net of the epidemic process in Example 2.

transitions with no incoming arcs are always enabled. A transition τ is fired by removing $l(a)$ tokens from every place with an arc a leading to t and adding $l(b)$ tokens to every place with an arc b coming from t. The firing of a transition results in a new marking m' and corresponds to a transition from m to m' in the underlying transition system. In an SPN the firing rate $\lambda_\tau > 0$ of transition τ determines the random delay during which τ has to be enabled before it can fire. The underlying graph is the intensity graph of a Markov chain if the transitions are labeled with their respective firing rates.

Example 2 (cont.) Fig. 2 shows a stochastic Petri net for the epidemic process. The net has two places x and y depicted as circles, and five transitions depicted as rectangles. (We omit the multiplicity labeling of the arcs because all arcs have multiplicity 1.) The firing rate of each transition is given by the transition label. Here, we use functions that depend on the current marking m. The initial marking m_0 is the empty marking, i.e., $m_0(x) = m_0(y) = 0$. □

3.3 Stochastic Process Algebras

Stochastic process algebras can be used to specify continuous-time Markov chains based on a high-level description language that emphasizes the construction of complex processes from simple processes [25, 19, 5, 36]. They provide several types of operators, such as prefix, choice, parallel composition, and recursion, in order to support different ways to combine processes. Typically, these languages are accompanied by structured operational semantics that defines a state-transition graph, whose states are process terms. The graph can then be transformed into the intensity graph of a Markov chain.

Originally, stochastic process algebras were designed to *explicitly* model different molecules of the same species, that is, the model distinguishes instances of components being in the same local state. Since for population models this may lead to an enormous blow-up of

the description, symmetry representations have been developed [23, 9]. They support the specification of the local state and number of instances for a component type. In this way, as in the other languages, the separate spatial identity of each molecule is hidden on the syntactical level.

Example 2 (cont.) In Bio-PEPA, the five reactions of the epidemic process are modeled as five actions (`r1` to `r5`). The two species are models as two sequential processes (`X` and `Y`) that synchronize on action `r5` in the model component. Below we show an input file for the Bio-PEPA workbench of the epidemic process.

```
r1 = [ a ];
r2 = [ b * X ];
r3 = [ c ];
r4 = [ d * Y ];
r5 = [ e * X * Y ];
X = r1>> + r2<< + r5<<;
Y = r3>> + r4<< + r5>>;
(X <r5> Y)
```

The first five lines specify the actions and the corresponding rates. Line 6 and 7 specify in which reactions the components take part and what role they play in the reaction. E.g., `r1>>`, which is a shortcut for `(r1,1) >> X`, means `X` is an reactant in reaction `r1` with stoichiometry coefficient 1 and `r2<<` means `X` is a product in reaction `r2`. The plus operator (`+`) is the sequential composition operator defining that the actions are sequentially interleaved. Finally, the last line specifies that the model is the parallel composition of processes `X` and `Y` that synchronize on reaction `r5`. □

3.4 Stoichiometric Equations

Markov chain models for networks of biochemical reactions are usually specified by means of stochiometric equations. A stochiometric equation describes a reaction type. For instance, $A + B \rightarrow C$ means that if a molecule of type A hits a molecules of type B, they may form a complex molecule C. We call the species that are consumed by a reaction *reactants*; in the above example, A and B are reactants.

Assume that the system involves n different chemical species $S_1, \ldots, S_n$. Consider a set $\{R_1, \ldots, R_m\}$ of chemical reactions, where the j-th reaction is given by the stochiometric equation

$$R_j : \quad l_{j,1}S_1 + \cdots + l_{j,n}S_n \;\rightarrow\; k_{j,1}S_1 + \cdots + k_{j,n}S_n.$$

The stoichiometric coefficients $l_{j,1}, \ldots, l_{j,n}$ and $k_{j,1}, \ldots, k_{j,n}$ are nonnegative integers and describe how many molecules of each type are consumed and produced by the reaction. In the equation, we may omit a species if its coefficient is 0, and we may omit coefficients that are 1. Assume that $x = (x_1, \ldots, x_n)$ is the current state of the system, that is, we have x_i molecules of species i in the system. If a reaction of type R_j occurs, then the successor state is $x + v_j$, where the *change vector* v_j is given by $v_j = (k_{j,1} - l_{j,1}, \ldots, k_{j,n} - l_{j,n})$.

For a given state x, an instance of reaction R_j may occur whenever there are enough reactants in the system, i.e., whenever all entries of the vector $x - (l_{j,1}, \ldots, l_{j,n})$ are nonnegative. In this case, there is a transition in the underlying state-transition graph between state x and state $x+v_j$.[1] The rate of reaction R_j in state x determines the corresponding transition label in the intensity graph.

Stochastic Chemical Kinetics. If the reaction R_j is an elementary reaction, meaning that each instance corresponds to a single mechanistic step, then the transition rate $\alpha_j(x)$ between state x and $x + v_j$ is given by

$$\alpha_j(x) = c_j \cdot \prod_{i=1}^{n} \binom{x_i}{l_{j,i}}, \tag{4}$$

where $c_j > 0$ is a constant. This definition reflects the law of mass action kinetics, which states that the rate at which a chemical reaction occurs is proportional to the product of the reactant concentrations. Stochastic chemical kinetics considers populations of chemical species and replaces the product of reactant concentrations by $\prod_{i=1}^{n} \binom{x_i}{l_{j,i}}$, which is the number of distinct reactant combinations [18]. Usually, the constant c_j appears above the reaction arrow in the stoichiometric equation.

[1] We assume for simplicity that each change vector v_j has at least one nonzero entry, and that all change vectors are distinct.

The following example shows that stoichiometric equations can also be used to describe population models from other application areas.

Example 2 (cont.) We describe the epidemic process as a network of the "reactions":

$$R_1 : \emptyset \xrightarrow{a} S_x \qquad R_2 : S_x \xrightarrow{b} \emptyset \qquad R_3 : \emptyset \xrightarrow{c} S_y$$
$$R_4 : S_y \xrightarrow{d} \emptyset \qquad R_5 : S_x + S_y \xrightarrow{e} 2S_y$$

Here, the symbol $\emptyset$ means that all stoichiometric coefficients are zero. Note that if the transition rates are are defined as in Eq. (4), they agree with the rates of Example 2. □

Since stoichiometric equations are classically used to describe biochemical reactions, we present an example from biology next.

Example 3. An enzyme-catalyzed substrate conversion is specified by the three reactions $R_1 : E + S \xrightarrow{c_1} ES$, $R_2 : ES \xrightarrow{c_2} E + S$, $R_3 : ES \xrightarrow{c_3} E + P$. This network involves four chemical species, namely, enzyme (E), substrate (S), complex (ES), and product (P) molecules. The change vectors are $v_1 = (-1, -1, 1, 0)$, $v_2 = (1, 1, -1, 0)$, and $v_3 = (1, 0, -1, 1)$. For $(x_1, x_2, x_3, x_4) \in \mathbb{Z}_+^4$, the rate functions are $\alpha_1(x_1, x_2, x_3, x_4) = c_1 \cdot x_1 \cdot x_2$, $\alpha_2(x_1, x_2, x_3, x_4) = c_2 \cdot x_3$, and $\alpha_3(x_1, x_2, x_3, x_4) = c_3 \cdot x_3$. □

Systems Biology Markup Language. For software tools in systems biology, a standard language for the specification of systems is the Systems Biology Markup Language (SBML) [27]. It is an XML-based format that describes biochemical reaction networks by a list of components. Each component may describe dynamic behaviors by reactions, events, and mathematical rules, or give details about reacting species or compartments. SBML also offers several mechanisms such as unit and parameter definitions to ensure the unambiguous understanding of quantitative descriptions.

Example 3 (cont.) In Fig. 3, we show a part of an SBML description of the enzyme-catalyzed substrate conversion. (The SBML description is taken from the SBML homepage [27].) Lines 22–27 define the species ES, P, S, and E. In lines 64–84, we can see the description of the reaction $R_3 : ES \xrightarrow{c_3} E + P$. Note that SBML uses an extended version of stoichiometric equations to describe reactions. Like a stoichiometric equation, every reaction has a set of reactants

and a set of products. However, the rate function is defined independently (cf. Fig. 3, lines 72–83) and need not follow Eq. (4). □

3.5 Guarded Commands

Similar to Petri nets, guarded-command models (GCM) describe the state transitions of the underlying process. However, unlike Petri nets, GCM are textual. Often, the set of all transitions can be partitioned into classes of transitions. Instead of listing all states, the modeler describes the possible classes of transitions that may occur. As a representative for such transition class description, we present a syntax that is inspired by Dijkstra's guarded-command language [14], which has subsequently been used by GCM such as Reactive Modules [1] and by the language for specifying PRISM models [12, 39]. We describe transition classes by guarded commands that operate on the state variables of the system. Recall that the state variables of the system are nonnegative integers representing numbers of molecules for each species. A guarded command takes the form

```
[] guard |- rate -> update
```

where the `guard` is a Boolean predicate over the variables, which determines in which states the corresponding transitions are enabled. The `update` is a rule that describes the change of the system variables if a corresponding transition is performed. Syntactically, `update` is a list of statements, each assigning to a variable an expression over variables. Assume that `x` is a variable. If, for instance, the update rule is that `x` is incremented by 1, we write `x:=x+1`. We assume that variables that are not listed in the update rule do not change if the transition is taken. Each guarded command also assigns a `rate` to the corresponding transitions, which is a function in the state variables. We do not fix an expression language for the rate functions here.

Example 2 (cont.) We define a GCM for the epidemic process.

```
variables x,y
[] true         |- a     -> x:=x+1
[] (x>0)        |- b*x   -> x:=x-1
[] true         |- c     -> y:=y+1
[] (y>0)        |- d*y   -> y:=y-1
[] (x>0)&(y>0)  |- e*x*y -> x:=x-1; y:=y+1
```

□

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <sbml level="2" version="3" xmlns="http://www.sbml.org/sbml/level2/version3">
 3   <model name="EnzymaticReaction">
 4     <listOfUnitDefinitions>
 5       <unitDefinition id="per_second">
 6         <listOfUnits>
 7           <unit kind="second" exponent="-1"/>
 8         </listOfUnits>
..       ...
19     <listOfCompartments>
20       <compartment id="cytosol" size="1e-14"/>
21     </listOfCompartments>
22     <listOfSpecies>
23       <species compartment="cytosol" id="ES" initialAmount="0"     name="ES"/>
24       <species compartment="cytosol" id="P"  initialAmount="0"     name="P"/>
25       <species compartment="cytosol" id="S"  initialAmount="1e-20" name="S"/>
26       <species compartment="cytosol" id="E"  initialAmount="5e-21" name="E"/>
27     </listOfSpecies>
28     <listOfReactions>
..       ...
64       <reaction id="R3" reversible="false">
65           <listOfReactants>
66             <speciesReference species="ES"/>
67           </listOfReactants>
68           <listOfProducts>
69             <speciesReference species="E"/>
70             <speciesReference species="P"/>
71           </listOfProducts>
72           <kineticLaw>
73             <math xmlns="http://www.w3.org/1998/Math/MathML">
74               <apply>
75                 <times/>
76                 <ci>cytosol</ci>
77                 <ci>c3</ci>
77                 <ci>ES</ci>
78               </apply>
79             </math>
80             <listOfParameters>
81               <parameter id="c3" value="0.1" units="per_second"/>
82             </listOfParameters>
83           </kineticLaw>
84       </reaction>
85     </listOfReactions>
86   </model>
87 </sbml>
```

Fig. 3. Part of the SBML description in XML syntax of an enzymatic reaction.

Note that each guarded command specifies infinitely many transitions. For examples, the guarded command `[] true |- a -> x:=x+1` specifies one transition from each state, with constant rate `a`, to a successor state in which the number of `x` molecules is incremented and the number of `y` molecules remains unchanged.

Example 3 (cont.) The enzyme reaction is specified by the guarded commands:

```
variables e,s,es,p
[] (e>0)&(s>0) |- c1*e*s -> e:=e-1;  s:=s-1; es:=es+1
[] (es>0)      |- c2*es -> es:=es-1; e:=e+1;  s:=s+1
[] (es>0)      |- c3*es -> es:=es-1; e:=e+1;  p:=p+1
```

□

Now, we show how to derive the underlying generator matrix from a GCM. To simplify the presentation, we assume that the updates of two commands differ whenever there is a state in which both guards are true. Moreover, we do not consider commands with empty updates, because "self-loops" do not alter the dynamics of a Markov chain. Then, each guarded command determines an entry in the row of a state s in the generator matrix whenever the guard is true in s. Assume that the state space of the underlying Markov chain is $S = \mathbb{Z}_+^n$, and $G \subseteq S$ is the subset where the guard is true. Furthermore, $s = (s_1, \ldots, s_n)$ and the update is a function $u : G \to S$. Then $q_{s,u(s)} = r(s)$, where $r : G \to \mathbb{R}_{\geq 0}$ is the rate function of the command. For instance, in Example 2, $G = \{(x, y) \in \mathbb{Z}_+^2 \mid x > 0 \text{ and } y > 0\}$ is the guard set of the last command. The update function is $u(x, y) = (x - 1, y + 1)$, and the rate function is $r(x, y) = e \cdot x \cdot y$, which yields the matrix entries $q_{(x,y),(x-1,y+1)} = e \cdot x \cdot y$ for all $x > 0$ and $y > 0$.

4 Properties of Specification Languages

In this section, we discuss several properties of specification languages which are important for the construction and the analysis of a model. We focus on the languages mentioned in the previous section.

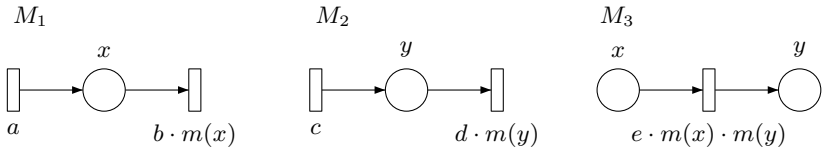

Fig. 4. Composition of stochastic Petri nets.

4.1 Compositionality

Compositionality facilitates the description of complex systems. A compositional language allows the modular description of a system by combining submodels that describe parts of the system. Moreover, a modular description can be advantageous for the analysis of the model, e.g., for compositional aggregation techniques [6].

Matrix Descriptions. We can construct the generator matrix Q of the epidemic process in Example 2 in a compositional way. We define the three matrices $A = \mathit{diag}([1\ 1\ 1\ldots], 1)$, $B = \mathit{diag}([0\ 1\ 2\ldots], 1)$, $C = \mathit{diag}([0\ 1\ 2\ldots], -1)$, where for $k \in \mathbb{Z}$ the notation $\mathit{diag}(v, k)$ refers to a matrix whose nonzero elements are the elements of the vector v that appears on the k-th diagonal of the matrix (negative values indicate that the vector appears below the main diagonal). Then the matrix

$$\hat{Q} = ((a \cdot A + b \cdot C) \oplus (c \cdot A + d \cdot C)) + e \cdot (C \otimes B)$$

agrees with Q except for the main diagonal. Thus, if $\mathbf{1}$ is the column vector with all entries equal to 1, then $Q = \hat{Q} - \mathit{diag}(\hat{Q} \cdot \mathbf{1}, 0)$. The matrix $\hat{Q}$ describes a network of two stochastic automata. The first automaton represents the state variable x, and $(a \cdot A + b \cdot C)$ defines its local transitions. The second automaton represents the state variable y, and $(c \cdot A + d \cdot C)$ defines its local transitions. Finally, $e \cdot (C \otimes B)$ describes the synchronous transitions of the network.

Thus, the composition of models with matrix representation requires matrix operations such as matrix sum, Kronecker product, and Kronecker sum.

Stochastic Petri Nets. In Fig. 4, we show the stochastic Petri nets of three subsystems of Example 2. Their combination yields the model shown in Fig. 2. The composition of Petri nets hinges on the identity of the places, because places with equal labels are collapsed. Thus, renaming of variables may be necessary. In the composite model, the original models are often not clearly separable.

Stochastic Process Algebras. Compositionality is one of the most important aspect of process calculi. Process algebras are equipped with various operators that can be used to combine process terms. This facilitates the description of different forms of interaction between subsystems. For instance, a detailed discussion on synchronous interaction, we refer to [24]. Note that in the Bio-PEPA example in the previous section, the epidemic process is the composition of the two process terms `X` and `Y`.

Stoichiometric Equations. As for Petri nets, sets of stoichiometric equations may be joined, where the interfaces are specified by the names of the chemical species. For instance, the composition of the three networks M_1, M_2, M_3 of reactions specified by

M_1	M_2	M_3
$R_1 : \emptyset \xrightarrow{a} S_x$	$R_1 : \emptyset \xrightarrow{c} S_y$	$R_1 : S_x + S_y \xrightarrow{e} 2S_y$
$R_2 : S_x \xrightarrow{b} \emptyset$	$R_2 : S_y \xrightarrow{d} \emptyset$	

yields the description of the epidemic process in Example 2. Here, the composite model is constructed simply by the union of reactions.

Guarded Commands. Again, we consider Example 2. The GCM for the subsystems we discussed above are

```
variables x
[] true        |- a     -> x:=x+1
[] (x>0)       |- b*x   -> x:=x-1

variables y
[] true        |- c     -> y:=y+1
[] (y>0)       |- d*y   -> y:=y-1

variables x,y
[] (x>0)*(y>0) |- e*x*y -> x:=x-1; y:=y+1
```

Similar as for stoichiometric equations, two GCM can be composed by a simple union of the guarded commands, where variables may have to be renamed.

4.2 Expressiveness and Succinctness

Two important properties of a specification language are its expressive power and its succinctness. For example, language A is as ex-

pressive as language B if every model that can be specified in B can also be specified in A. Of two equally expressive language, one may be more succinct than the other. For instance, if for some models there are descriptions in A that are exponentially smaller than all descriptions in B, then on these models, A is exponentially more succinct than B. The expressiveness and succinctness of specification languages can be compared by studying translations between languages and the cost of such translations. Other questions that fall under this topic concern the ease of extending a language to gain expressive power, and an independent characterization of which semantic objects (i.e., continuous-time Markov chains that arise from population models) can be described by expressions within a given formal syntax.

To our knowledge, no systematic and complete comparison between the various languages for describing population models has been carried out, and we make here only a few remarks.

Matrix Descriptions. The expressive power of a matrix description depends on the exact syntax of expressions for describing matrix entries and, in the case of infinite dimension, on the syntax for describing sets of entries. As first step in an analysis procedure, many other languages for specifying population models are translated into matrix descriptions. The translation from higher-level languages such as guarded commands often results in a blow-up of the size of the description.

Stochastic Petri Nets. A stochastic Petri net can be transformed into a guarded command model if we associate a variable with each place and a guarded command with each transition. Many extensions of stochastic Petri nets have been developed, such as generalized stochastic Petri nets [31], fluid stochastic Petri nets [26], etc. They can be used to describe stochastic processes that are not necessarily Markov chains. There is, however, we know of no extension of Petri nets that can specify infinitely branching Markov chains, whose intensity graph contains states with an infinite number of out-going transitions. Moreover, even though firing rates may be marking dependent, the expression syntax may not allow arbitrary rate functions.

Stochastic Process Algebras. Stochastic process algebras such as PEPA [25], TIPP [19], EMPA [5] and the stochastic pi-calculus [36] consider constant transition rates, i.e., transition rates do not depend on the state variables. This is because these languages originally were not designed for specifying population models. The extension Bio-PEPA [9] addresses this shortcoming, as was shown in the examples discussed in the previous section. Although most stochastic process algebras provide limited support for rate functions, they have recursion operators for specifying Markov chains for which no basic guarded command model can be constructed. The usefulness of such operators, however, depends on the application area.

Stoichiometric Equations. While stoichiometric equations are widely used to model networks of biochemical reactions, some models —such as the bistable toggle switch shown below— have rate functions that differ from Eq. (4). Thus, they cannot be described in the classical stoichiometric style.

Example 4. The bistable toggle switch is a prototype of a genetic switch with two competing repressor proteins and four reactions [17, 45]. It involves two chemical species, A and B, and four reactions. The reactions are $\emptyset \to A$, $A \to \emptyset$, $\emptyset \to B$, and $B \to \emptyset$. Let $x = (x_1, x_2) \in \mathbb{N}_0^2$. The rate functions are $\alpha_1(x) = c_1/(c_2 + x_2^2)$, $\alpha_2(x) = c_3 \cdot x_1$, $\alpha_3(x) = c_4/(c_5 + x_1^2)$, and $\alpha_4(x) = c_6 \cdot x_2$. Here, the values c_1, c_2, c_4, and c_5 are positive constants that determine the mutual repression of A and B. The values c_3 and c_6 are positive constants that determine at which rate degradation of molecules occurs. The toggle switch can be specified using SBML syntax, as SBML provides more flexibility than stoichiometric equations. □

A set of stoichiometric equations can always be transformed into a guarded command model or a stochastic Petri net. The number of chemical species corresponds to the number of variables (or places), and each reaction induces a guarded command (or transition).

Guarded Commands. A guarded command model can be transformed into a stochastic Petri net, where each command corresponds to a transition and each variable to a place. Note that this transformation is only possible if the guards are lower bounds on the state variables. Guarded command models are similar to stochastic Petri nets in that they allow rate functions that depend on the state vari-

ables. For instance, we can describe Example 4 using the following commands:

```
variables x1,x2
[] true |- c1/(c2+x2^2) -> x1:=x1+1
[] x1>0 |- c3*x1        -> x1:=x1-1
[] true |- c4/(c5+x1^2) -> x2:=x2+1
[] x2>0 |- c6*x2        -> x2:=x2-1
```

The limitations of basic guarded command models are similar to those of stochastic Petri nets; for example, infinitely branching Markov chains cannot be described by specifications consisting of finitely many guarded commands.

4.3 Executability

For the analysis of a model it is important that we can easily compute the direct transition successors of a given state from the model description. This allows us to "execute" the model, by repeatedly applying the next-state function [16].

Stochastic Petri Nets and Stochastic Process Algebras. Petri net and process algebra models provide a high-level description that is usually not directly executable. For a given marking in a stochastic Petri net, we have to inspect each place and each arc in order to determine the enabled transitions as well as their firing rates. Similarly, for a given stochastic process term, we have to consider each subterm and compute all possible transitions and their rates. Then, the composition rules determine the possible global transitions of the system and their rates. Therefore, most tools for the analysis of Markov chains with high-level specification languages construct an intermediate low-level model, such as a matrix representation [8, 4, 30, 29].

Guarded Commands and Stoichiometric Equations. For languages based on transition classes, such as guarded commands, the construction of an intermediate low-level model is not necessary. This i one of the main strengths of guarded commands. For a given state, an on-the-fly calculation of all possible successor states and transition rates can be performed by iterating over the set of guarded commands and calling their update and rate functions [22]. Note that for stoichiometric equations we have to store for each reaction

R_j the change vector v_j, the rate function α_j, and the number of necessary reactant molecules. This is essentially the representation provided by guarded commands.

4.4 Well-formedness

In Section 3, we discussed different formalisms for the specification of a Markov chain. In most languages, however, it is possible to specify models whose underlying intensity graph (or generator matrix) do not uniquely determine a Markov chain. This is due to the fact that the Kolmogorov backward equations (see Eq. (1)) may not have a unique solution. In Section 2, we used the following sufficient but not necessary condition:

$$\sup_{i \in S} |q_{ii}| < \infty \tag{5}$$

If the modeling formalism allows us to specify transition rates that are functions in the state variables, then this condition may not be fulfilled. Note that this can only occur if the number of reachable states is infinite. For instance, the condition in Eq. (5) is not satisfied in Example 2. In the sequel we discuss conditions that are weaker than Eq. (5) but still ensure a unique solution to Eq. (1). We focus on conditions on the matrix Q and refer to the entries of Q as q_{ij}. For the nonpositive diagonal entries, we use the abbreviation $q_i = -q_{ii}$.

A generator matrix Q is called *stable* if all entries are finite, and *conservative*, if all rows sum up to zero [3]. In most of the specification languages, stability and conservation of the underlying generator matrices are guaranteed by construction. Only models specified as matrix descriptions have to be checked separately.

A conservative and stable generator matrix that has a unique solution to the Kolmogorov backward equations is called *regular*. The following criterion is sufficient and necessary for a generator matrix to be regular.

Theorem 1 (Reuter's Criterion). *A stable and conservative generator Q on S is regular if and only if for any real $\lambda > 0$, the system of equations*

$$\sum\nolimits_{j \in S, j \neq i} q_{ij} z_j = (\lambda + q_i) z_i \text{ for all } i \in S \tag{6}$$

admits no nonnegative bounded solution other than the trivial one.

Example 5. Consider a model with the following guarded command.

```
x>0 |- 2^x -> x:=x+1
```

Recall that this model specifies a generator matrix Q with the following non-zero entries: $q_{i(i+1)} = 2^i$ and $q_{ii} = -2^i$ for all $i > 0$. Then, we obtain for Q the following equations from (6):

$$2^i z_{i+1} = (\lambda + 2^i) z_i \text{ for all } i > 0.$$

Applying simple transformations allows us to express the solutions for $i > 1$ in terms of z_1 by $z_i = \Pi_{k=1}^{i-1}(\frac{\lambda}{2^k}+1)z_1$. We choose $\lambda = z_1 = 1$, then it remainds to show that $z_i = \Pi_{k=1}^{i-1}(\frac{1}{2^k} + 1)$ is bounded for all $i > 1$. Since $z_i = e^{\ln(z_i)}$, it suffices to show that $\ln(\Pi_{k=1}^{i-1}(\frac{1}{2^k} + 1)) = \sum_{k=1}^{i-1} \ln(\frac{1}{2^k} + 1) \leq \sum_{k=1}^{i-1} \frac{1}{2^k} \leq \frac{1}{1-1/2} - 1 = 1$ is bounded for all $i > 1$. This shows that our model has a nontrivial bounded solution and we can conclude that Q is not regular. □

Since showing that Reuter's criterion is true for a model is rather difficult, we discuss another condition that is sufficient but not necessary. The idea is to approximate the infininte unbounded generator matrix Q by a sequence of bounded submatrices.

Theorem 2 ([3] Corollary 2.16). *Let Q be a conservative generator matrix over the state space S and let $S_1, S_2, \ldots$ be a sequence of subsets of S such that $S_1 \subseteq S_2 \subseteq \ldots$, $\cup_{r=1}^{\infty} S_r = S$, and $\sup_{i \in S_r} q_i < \infty$. Suppose that $z_j \geq 0$, $j \in S$ are such that*

1. *$\lim_{r \to \infty} \inf_{j \notin S_r} z_j = \infty$, and*
2. *there $\lambda \in \mathbb{R}$ such that $\sum_{j \neq i} q_{ij} z_j \leq (\lambda + q_i) z_i$ for all $i \in S$.*

Then Q is regular.

Given a suitable sequence S_r and values z_j, the sum and the number of conditions we have to check for regularity are infinite. The chosen syntax for the model description may facilitate the regularity check. For instance, in a GCM, it suffices to let the sum range over the finite set of guarded commands. We can use the guards to partition the state space into a finite number of sets such that we have to check a single condition for each set.

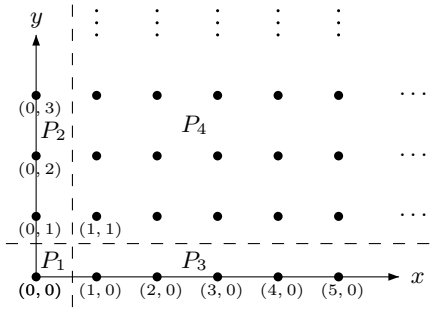

Fig. 5. State space partitioning w.r.t. the guarded command model of the epidemic process.

Example 2 (cont.) Recall the GCM of the epidemic process.

```
variables x,y
[] true         |- a     -> x:=x+1
[] (x>0)        |- b*x   -> x:=x-1
[] true         |- c     -> y:=y+1
[] (y>0)        |- d*y   -> y:=y-1
[] (x>0)&(y>0) |- e*x*y -> x:=x-1; y:=y+1
```

We use the four guards to partition the state space into four sets P_1, P_2, P_3, P_4, as shown in Fig. 5 by the dashed lines. Note that for suitable expression languages for the guards, we can always find this partitioning automatically. Now, for each set, we can check Theorem 2 using the corresponding guarded commands. For instance, in P_3 (right lower corner in Fig. 5), the first three guarded commands are enabled, and the second condition in Theorem 2 rewrites to $a{\cdot}f(x{+}1,y){+}b{\cdot}x{\cdot}f(x{-}1,y){+}c{\cdot}f(x,y{+}1) \leq (\lambda{+}a{+}b{\cdot}x{+}c){\cdot}f(x,y)$, where the functions f refers to the values z_i, $i \in S$. With $f(x,y) = x+y+1$ and $S_r = \{(x,y) \mid f(x,y) \leq r\}$, which satisfy Condition 1 in Theorem 2, we obtain $a\cdot(x{+}y{+}2){+}b\cdot x\cdot(x{+}y){+}c\cdot(x{+}y{+}2) \leq (\lambda{+}a{+}b{\cdot}x{+}c){\cdot}(x{+}y{+}1)$, which is the same as $a{-}b{\cdot}x{+}c \leq \lambda{\cdot}(x{+}y{+}1)$ and true for $(x,y) \in P_3$ if $\lambda = a + c$. □

In a similar way, it is possible to exploit the chosen syntax in order to decide whether the limit probabilities $\pi_i = \lim_{t\to\infty} p_i(t)$ of the Markov chain form a distribution. We refer to [3] for criteria that ensure the existence of a limit distribution as well as that it can be calculated according to Eq. (3).

In summary, GCM offer an efficiently checkable sufficient condition for regularity and the existence of the limit probabilities.

5 Analysis of Continuous-time Markov Chains

For the analysis of Markov chains we distinguish transient and steady-state analysis. The former refers to the computation of the vector $p(t)$ which contains the probabilities $P(X(t) = x)$ for each state x reachable from a given initial distribution. Often, $p(t)$ is computed at several time instances t of interest. Steady-state analysis requires the computation of the limiting behaviour of the Markov chain, i.e., of the probability vector $\lim_{t\to\infty} p(t)$.

There are three different approaches to the analysis of continuous-time Markov chains, namely, analytical solutions, numerical solutions, and simulation. Since analytical solutions can only be obtained for Markov chains with a very simple structure, we concentrate on the latter two approaches. Numerical solutions are based on an exploration of the state space which proceeds in a breadth-first search manner, by moving the probability mass through the state space. In contrast, simulation of Markov chains is a special case of the Monte Carlo method, which relies on the repeated generation of random sample paths in the underlying state-transition graph.

5.1 Simulation

Monte-Carlo simulation of Markov chains is based on the idea of generating a number of trajectories $X^{(t)}(\omega)$ using pseudo-random numbers [2]. Then, probabilities and expectations of certain random variables can be statistically estimated. The main advantage of simulation is that the memory requirements are low and therefore the analysis of systems of arbitrary size is possible. In order to achieve a high accuracy, however, a large number of trajectories have to be generated, which is very time consuming and often infeasible [13].

For the generation of trajectories, an executable model description is advantageous, as this will speed up the time needed for constructing each step of a trajectory.

5.2 Numerical Solutions

Several tools exist that provide algorithms for the numerical solution of continuous-time Markov chains [30, 4, 29, 8]. They use a matrix description with is obtained from a high-level modeling formalism

such as stochastic Petri nets [4, 8], guarded command languages [30, 29], or stochastic process algebra [30, 29]. The generator matrix of the Markov chain is stored either symbolically using multi-terminal BDDs [10] or sparse matrix packages are used. Moreover, if the size of the generator matrix exceeds the available memory capacity the matrix is stored as a Kronecker product of smaller matrices.

The implemented algorithms for the computation of $p(t)$ are either based on the solution of Eq. (2) or a discretization of the Markov chain. In the former case, numerical integration methods or methods based on a Krylov subspace construction are applied [42]. They require the construction of the generator matrix Q, which is often infeasible for large Markov chains. It is, however, possible to exploit the structure of the Markov chain and approximate the solution by successively considering submatrices of Q [22].

During the discretization procedure, also called uniformization [28, 20], a discrete-time Markov chain is constructed which has essentially the same transition graph structure as the original continuous-time Markov chain. The idea behind uniformization is that the maximum of the diagonal entries of the generator matrix can be used to "normalize" the time that the process remains in a state. Thus, an a-priori exploration of the state space is necessary to apply uniformization. For large Markov chains the memory requirements are can be prohibitive, even if sparse matrix structures or symbolic representations are used. Therefore variants of the uniformization method have been developed which exploit a Kronecker representation of the matrix [7].

Most formalisms allow us to specify an infinite number of reachable states, but are limited to finite models for their numerical analysis, because the analysis requires the enumeration of all reachable states and the construction of the generator matrix. An algorithm that completely avoids the construction of any matrix and exploits a guarded command description has been proposed recently in [13]. It can be used for the approximate analysis of infinite-state systems and does not suffer from excessive memory requirements, at least for systems where the significant part of the probability mass is concentrated on a manageable subset of states. The proposed algorithm is enhanced by the executability properties of the guarded command description.

For the computation of steady-state measures, most methods also require the a-priori construction of the generator matrix from the Markov chain specification. If the state space is large, direct methods for the solution of Eq. (3) are inefficient, and iterative methods such as the Jacobi, Gauss-Seidel, or SOR method must be applied [42]. Many iterative methods have been adapted so that they exploit a Kronecker representations of the generator matrix [43, 33]. Other approaches are based on on-the-fly techniques [11] or reduced state spaces that are obtained by exploiting symmetries in the model structure and tailoring to the variable in question [40].

6 Conclusions

There are many different languages for describing Markov chains with continuous time. The choice of an appropriate syntax usually depends on the application area. Guarded commands provide a natural language for the description of population models. They facilitate the specification of such models, because they are compositional, succinct, and provide sufficient expressive power. Moreover, they support well-formedness checks and allow a direct execution of the model.

References

1. R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
2. W. K. A.M. Law. *Simulation Modeling and Analysis*. McGraw Hill, 2000. 3rd ed.
3. W. Anderson. *Continuous-time Markov chains: An applications-oriented approach.* Springer, 1991.
4. S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis. The greatspn tool: recent enhancements. *SIGMETRICS Perform. Eval. Rev.*, 36(4):4–9, 2009.
5. M. Bernardo and R. Gorrieri. Extended Markovian process algebra. In *Proc. CONCUR 1996*, number 1119 in LNCS, pages 315–330. Springer, 1996.
6. P. Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of applied probability*, 31(1):59–75, 1994.
7. P. Buchholz and W. H. Sanders. Approximate computation of transient results for large markov chains. In *Proc. of QEST'04*, pages 126–135. IEEE Computer Society, 2004.
8. G. Ciardo, R. L. Jones, III, A. S. Miner, and R. I. Siminiceanu. Logic and stochastic modeling with smart. *Perform. Eval.*, 63(6):578–608, 2006.

9. F. Ciocchetta and J. Hillston. Bio-pepa: a framework for modelling and analysis of biological systems. *Theoretical Computer Science*, 2009. To appear.
10. E. Clarke, M. Fujita, P. McGeer, J. Yang, and X. Zhao. Multi-terminal binary decision diagrams: An ecient data structure for matrix representation. In *In Proc. IWLS93*, 1993.
11. D. D. Deavours and W. H. Sanders. "On-the-fly" solution techniques for stochastic Petri nets and extensions. In *IEEE TSE*, pages 132–141, 1997.
12. P. development team. The prism language.
13. F. Didier, T. Henzinger, M. Mateescu, and V. Wolf. Approximation of event probabilities in noisy cellular processes. In *Proc. of CMSB*, LNCS. Springer, 2009. to appear.
14. E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975.
15. P. Fernandes, B. Plateau, and W. J. Stewart. Numerical evaluation of stochastic automata networks. In *Proc. of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pages 179–183, 1995.
16. J. Fisher and T. A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25:1239–1249, 2007.
17. T. Gardner, C. Cantor, and J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403:339 – 342, 2000.
18. D. T. Gillespie. *Markov Processes*. Academic Press., 1992.
19. N. Götz, U. Herzog, and M. Rettelbach. Tipp – a language for timed processes and performance evaluation. report Technical Report 4/92, IMMD VII, University of Erlangen-Nurnberg, 1992.
20. D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):926–944, 1984.
21. P. J. Haas. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer, 2002.
22. T. Henzinger, M. Mateescu, and V. Wolf. Sliding window abstraction for infinite Markov chains. In *Proc. CAV*, LNCS. Springer, 2009. To appear.
23. H. Hermanns. An operator for symmetry representation and exploitation in stochastic process algebras. In *Proc. of PAPM 97*, pages 55–70, 1997.
24. J. Hillston. The nature of synchronisation. In *Proc. of PAPM'94*, pages 51–70, 1994.
25. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
26. G. Horton, V. G. Kulkarni, D. M. Nicol, and K. S. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1):184–201, 1998.
27. M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, and H. Kitano. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *BIOINFORMATICS*, 19(4):524–531, 2003.
28. A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.
29. J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov reward model checker. In *Proc. of QEST'05*, pages 243–244. IEEE Computer Society, 2005.
30. M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

31. M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. Modelling with generalized stochastic petri nets. *SIGMETRICS Perform. Eval. Rev.*, 26(2):2, 1998.
32. H. H. McAdams and A. Arkin. It's a noisy business! *Trends in Genetics*, 15(2):65–69, 1999.
33. T. D. P. Buchholz. Block SOR preconditioned projection methods for Kronecker structured Markovian representations. *SIAM Journal on Scientific Computing*, 26(4):1289–1313, 2005.
34. J. Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–418, 2004.
35. B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. of the Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 147–154, 1985.
36. C. Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.
37. C. Rao, D. Wolf, and A. Arkin. Control, exploitation and tolerance of intracellular noise. *Nature*, 420(6912):231–237, 2002.
38. G. E. H. Reuter. Competition processes. In *In Proc. 4th Berkeley Symp. Math. Statist. Prob.*, volume 2, pages 421–430. University of California Press, Berkeley, 1961.
39. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.
40. W. H. Sanders, E. Y, and J. Meyer. Reduced base model construction methods for stochastic activity networks. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, volume 11, pages 74 – 84, 1989.
41. R. Srivastava, L. You, J. Summers, and J. Yin. Stochastic vs. deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology*, 218:309–321, 2002.
42. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1995.
43. W. J. Stewart, K. Atif, and B. Plateau. The numerical solution of stochastic automata networks. *European Journal of Operational Research*, 86(3):503–525, 1995.
44. P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Science, USA*, 99(20):12795–12800, 2002.
45. T. Tian and K. Burrage. Stochastic models for regulatory networks of the genetic toggle switch. *Proc. Natl. Acad. Sci. USA*, 103(22):8372–8377, May 2006.
46. T. E. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modelling in vivo reactions. *Computational Biology and Chemistry*, 28:165–178, 2004.
47. D. J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.