Contents

Chapter 2

Background

2.1 Petri nets

Petri nets are a widely used graphical and mathematical modeling tool for systems which are concurrent, asynchronous, distributed, parallel or nondeterministic.

Definition 2.1 A *Petri net* is a 5-tuple $PN = (P, T, F, W, M_0)$, where
- $P = \{p_0, p_1, \ldots, p_{n-1}\}$ is a finite set of places;
- $T = \{t_0, t_1, \ldots, t_{m-1}\}$ is a finite set of transitions;
- $F \subseteq (P \times T) \cup (P \times T)$ is a set of arcs, also called the flow relation;
- $W : F \to \mathbb{N}^+$ is an arc weight function;
- $M_0 : P \to \mathbb{N}$ is the initial marking;
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$ [33].

Arcs from P to T are called *input arcs*. The input places of a transition t are denoted by ${}^\bullet t = \{p : (p, t) \in F\}$. In contrast, arcs of the form (t, p) are called *output arcs* and the output places of t are denoted by $t^\bullet = \{p : (t, p) \in F\}$.

A *marking* $M : P \to \mathbb{N}$ assigns a number of *tokens* to each place. The transition t is *enabled* in the marking M (written as $M[t\rangle$) when $M(p) \geq W(p, t)$ for all $p \in {}^\bullet t$.

Petri nets are graphically represented as edge weighted directed bipartite graphs. Places are drawn as circles, while transitions are drawn as bars or rectangles. Edge weights of 1 are ususally omitted from presentation. Dots on places correspond to tokens in the current marking.

If $M[t\rangle$ the transition t can be *fired* to get a new marking M' (written as $M[t\rangle M'$) by decreasing the token counts for each place $p \in {}^\bullet t$ by $W(p, t)$ and increasing the token counts for each place $p \in t^\bullet$ by $W(t, p)$. Note that in general, ${}^\bullet t$ and $t^\bullet$ need not

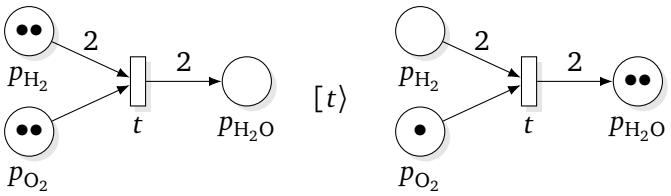


Figure 2.1 A Petri net model of the reaction of hydrogen and oxygen.

be disjoint. Thus, the firing rule can be written as

$$M'(p) = M(p) - W(p,t) + W(t,p), \tag{2.1}$$

where we take $W(x,y) = 0$ if $(x,y) \notin F$ for brevity.

A marking M' is *reachable* from the marking M (written as $M \rightsquigarrow M'$) if there exists a sequence of markings and transitions for some finite k such that

$$M = M_1 [t_{i_1}\rangle M_2 [t_{i_2}\rangle M_3 [t_{i_3}\rangle \cdots [t_{i_{k-1}}\rangle M_{k-1} [t_{i_k}\rangle M_k = M'.$$

A marking M is in the reachable *state space* of the net if $M_0 \rightsquigarrow M$. The set of all markings reachable from M_0 is denoted by

$$RS = \{M : M_0 \rightsquigarrow M\}.$$

Definition 2.2 The Petri net PN is k-*bounded* if $M(p) \leq k$ for all $M \in RS$ and $p \in P$. PN is *bounded* if it is k-bounded for some (finite) k.

The reachable state space RS is finite if and only if the Peti net is bounded.

Example 2.1 The Petri net in Figure 2.1 models the chemical reaction

$$2\,H_2 + O_2 \rightarrow 2\,H_2O.$$

In the initial marking (left) there are two hydrogen and two oxygen molecules, represented by tokens on the places p_{H_2} and p_{O_2}, therefore the transition t is enabled. Firing t yields the marking on the right where the two tokens on p_{H_2O} are the reaction products. Now t is no longer enabled.

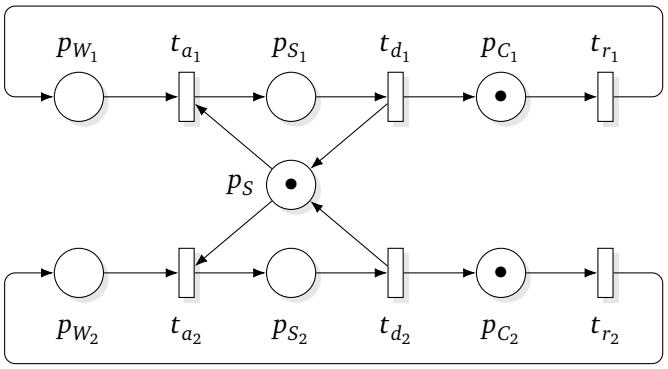


Figure 2.2 The *SharedResource* Petri net model.

Running example 2.2 In Figure 2.2 we introduce the *SharedResource* model which will serve as a running example throughout this report.

The model consists of a single shared resource S and two consumers. Each consumer can be in one of the C_i (calculating locally), W_i (waiting for resource) and S_i (using shared resource) states. The transitions r_i (request resource), a_i (acquire resource) and d_i (done) correspond to behaviors of the consumers. The net is 1-bounded, therefore it has finite RS.

The Petri net model allows the verification of safety properties, e.g. we can show that there is mutual exclusion – $M(S_1) + M(S_2) \leq 1$ for all reachable markings – or that deadlocks cannot occur. In contrast, we cannot compute dependability or performability measures (e.g. the utilization of the shared resource or number of calculations completed per unit time) because the model does not describe the temporal behavior of the system.

2.1.1 Petri nets extended with inhibitor arcs

One of the most frequently used extensions of Petri nets is the addition of inhibitor arcs, which constrains the rule for transition enablement. This modification gives Petri nets expressive power equivalent to Turing machines [10].

Definition 2.3 A *Petri net with inhibitor arcs* is a 3-tuple $PN_I = (PN, I, W_I)$, where

- $PN = (P, T, F, W, M_0)$ is a Petri net;
- $I \subseteq P \times T$ is the set of inhibitor arcs;
- $W_I : I \to \mathbb{N}^+$ is the inhibitor arc weight function.

Let ${}^\circ t = \{p : (p,t) \in I\}$ denote the set of inhibitor places of the transition t. The enablement rule for Petri nets with inhibitor arcs can be formalized as

$$M[t\rangle \iff M(p) \geq W(p,t) \text{ for all } p \in {}^\bullet t \text{ and } M(p) < W_I(p,t) \text{ for all } p \in {}^\circ t.$$

The firing rule (2.1) remains unchanged.

2.2 Continuous-time Markov chains

Continuous-time Markov chains are mathematical tools for describing the behavior of systems in countinous time where the random behavior of the system only depends on its current state.

Definition 2.4 A *Continuous-time Markov Chain* (CTMC) $X(t) \in S, t \geq 0$ over a finite or countable infinite state space $S = \{0, 1, \ldots, n-1\}$ is a continuous-time random process with the *Markovian* or memoryless property

$$\begin{aligned}\mathbb{P}(X(t_k) = x_k \mid X(t_{k-1}) = x_{k-1}, X(t_{k-2}) = x_{k-2}, \ldots, X(t_0) = x_0) \\ = \mathbb{P}(X(t_k) = x_k \mid X(t_{k-1}) = x_{k-1}),\end{aligned}$$

where $t_0 \leq t_1 \leq \cdots \leq t_k$. A CTMC is said to be *time-homogenous* if it also satisfies

$$\mathbb{P}(X(t_k) = x_k \mid X(t_{k-1}) = x_{k-1}) = \mathbb{P}(X(t_k - t_{k-1}) = x_k \mid X(0) = x_{k-1}),$$

i.e. it is invariant to time shifting.

In this report we will restrict our attention to time-homogenous CTMCs over finite state spaces. The state probabilities of these stochastic processes at time t form a finite-dimensional vector $\boldsymbol{\pi}(t) \in \mathbb{R}$,

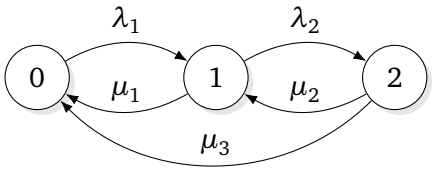
$$\pi(t)[x] = \mathbb{P}(X(t) = x)$$

that satisfies the differential equation

$$\frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t} = \boldsymbol{\pi}(t)Q \tag{2.2}$$

for some square matrix Q. The matrix Q is called the *infinitesimal generator matrix* of the CTMC and can be interpreted as follows:

- The diagonal elements $q[x,x] < 0$ describe the holding times of the CTMC. If $X(t) = x$, the *holding time* $h_x = \inf\{h > 0 : X(t+h) \neq x\}$ spent in state x is exponentially distributed with rate $\lambda_x = -q[x,x]$. If $q[x,x] = 0$, then no transitions are possible from state x and it is said to be *absorbing*.



$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} -\lambda_1 & \lambda_1 & 0 \\ \mu_1 & -\lambda_2 - \mu_1 & \lambda_2 \\ \mu_3 & \mu_2 & -\mu_2 - \mu_3 \end{pmatrix} \end{array}$$

Figure 2.3 Example CTMC with 3 states and its generator matrix.

- The off-diagonal elements $q[x, y] \geq 0$ describe the state transitions. In state x the CTMC will jump to state y at the next state transition with probability $-q[x, y]/q[x, x]$. Equivalently, there is expontentially distributed countdown in the state x for each $y : q[x, y] > 0$ with *transition rate* $\lambda_{xy} = q[x, y]$. The first countdown to finish will trigger a state change to the corresponding state y. Thus, the CTMC is a transition system with exponentially distributed timed transitions.

- Elements in each row of Q sum to 0, hence it satisfies $Q\mathbf{1}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$.

For more algebraic properties of infinitesimal generator matrices, we refer to Plemmons and Berman [35] and Stewart [43].

A state y is said to be *reachable* from the state x ($x \rightsquigarrow y$) if there exists a sequence of states

$$x = z_1, z_2, z_3, \ldots, z_{k-1}, z_k = y$$

such that $q[z_i, z_{i+1}] > 0$ for all $i = 1, 2, \ldots, k-1$. If y is reachable from x for all $x, y \in S$ y, the Markov chain is said to be *irreducible*.

The *steady-state probability distribution* $\boldsymbol{\pi} = \lim_{t\to\infty} \boldsymbol{\pi}(t)$ exists and is independent from the *initial distribution* $\boldsymbol{\pi}(0) = \boldsymbol{\pi}_0$ if and only if the finite CTMC is irreducible. The steady-state distribution is a stationary solution of eq. (2.2), therefore it satisfies the linear equation

$$\frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}t} = \boldsymbol{\pi}Q = \mathbf{0}, \quad \boldsymbol{\pi}\mathbf{1}^{\mathrm{T}} = 1. \tag{2.3}$$

Example 2.3 Figure 2.3 shows a CTMC with 3 states. The transitions from state 0 to 1 and from 1 to 2 are associated with exponentially distributed countdowns with rates λ_1 and λ_2 respectively, while transitions in the reverse direction have rates μ_1 and μ_2. The transition form state 2 to 0 is also possible with rate μ_3.

The rows (corresponding to source states) and columns (destination states) of the infinitesimal generator matrix Q are labeled with the state numbers. The diagonal element $q[1, 1]$ is $-\lambda_2 - \mu_1$, hence the holding time in state 1 is exponentially distributed with rate $\lambda_2 + \mu_1$. The transition to 0 is taken with probability

$-q[1,0]/q[1,1] = \mu_1/(\lambda_2+\mu_1)$, while the transition to 2 is taken with probability $\lambda_2/(\lambda_2+\mu_1)$.

The CTMC is irreducible, because every state is reachable from every other state. Therefore, there is a unique steady-state distribution $\boldsymbol{\pi}$ independent from the initial distribution $\boldsymbol{\pi}_0$.

2.2.1 Markov reward models

Continuous-time Markov chains may be employed in the estimation of performance measures of models by defining *rewards* that associate *reward rates* with the states of a CTMC. The momentary reward rate random variable $R(t)$ can describe performance measures defined at a single point of time, such as resource utilization or probability of failure, while the *accumulated reward* random variable $Y(t)$ may correspond to performance measures associated with intervals of time, such as total downtime.

Definition 2.5 A *Continuous-time Markov Reward Process* over a finite state space $S = \{0, 1, \ldots, n-1\}$ is a pair $(X(t), \mathbf{r})$, where $X(t)$ is a CTMC over S and $\mathbf{r} \in \mathbb{R}^n$ is a *reward rate vector*.

The element $r[x]$ of the reward vector is a momentary reward rate in state x, therefore the reward rate random variable can be written as $R(t) = r[X(t)]$. The accumulated reward until time t is defined by

$$Y(t) = \int_0^t R(\tau)\,\mathrm{d}\tau.$$

The computation of the distribution function of Y(t) is a computationally intensive task (a summary is available at [36, Table 1]), while its mean, $\mathbb{E}\,Y(t)$, can be computed efficiently as discussed below.

Given the initial probability distribution vector $\boldsymbol{\pi}(0) = \boldsymbol{\pi}_0$ the expected value of the reward rate at time t can be calculated as

$$\mathbb{E}\,R(t) = \sum_{i=0}^{n-1} \pi(t)[i]r[i] = \boldsymbol{\pi}(t)\,\mathbf{r}^{\mathrm{T}}, \tag{2.4}$$

which requires the solution of the initial value problem [22, 39]

$$\frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t} = \boldsymbol{\pi}(t)\,Q, \quad \boldsymbol{\pi}(0) = \boldsymbol{\pi}_0$$

to form the inner product $\mathbb{E}\,R(t) = \boldsymbol{\pi}(t)\,\mathbf{r}^{\mathrm{T}}$. To obtain the expected steady-state reward rate (if it exists) the linear equation (2.3) should be solved instead for the steady-state probability vector $\boldsymbol{\pi}$.

The expected value of the accumulated reward is

$$\begin{aligned}\mathbb{E}\,Y(t) &= \mathbb{E}\left[\int_0^t R(\tau)\,\mathrm{d}\tau\right] = \int_0^t \mathbb{E}[R(\tau)]\,\mathrm{d}\tau \\ &= \int_0^t \sum_{i=0}^{n-1} \pi(\tau)[i] r[i]\,\mathrm{d}\tau = \sum_{i=0}^{n-1} \int_0^t \pi(\tau)[i]\,\mathrm{d}\tau\, r[i] \\ &= \int_0^t \boldsymbol{\pi}(t)\,\mathrm{d}\tau\,\mathbf{r}^{\mathrm{T}} = \mathbf{L}(t)\,\mathbf{r}^{\mathrm{T}},\end{aligned}$$

where $\mathbf{L}(t) = \int_0^t \boldsymbol{\pi}(t)\,\mathrm{d}\tau$ is the accumulated probability vector, which is the solution of the initial value problem [39]

$$\frac{\mathrm{d}\mathbf{L}(t)}{\mathrm{d}t} = \boldsymbol{\pi}(t), \quad \frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t} = \boldsymbol{\pi}(t)Q, \quad \mathbf{L}(0) = \mathbf{0}, \quad \boldsymbol{\pi}(0) = \boldsymbol{\pi}_0.$$

Example 2.4 Let c_0, c_1 and c_2 denote operating costs per unit time associated with the states of the CTMC in Figure 2.3. Consider the Markov reward process $(X(t), \mathbf{r})$ with reward rate vector

$$\mathbf{r} = \begin{pmatrix} c_0 & c_1 & c_2 \end{pmatrix}.$$

The random variable $R(t)$ describes the momentary operating cost, while $Y(t)$ is the total operating expenditure until time t. The steady-state expectation of R is the average maintenance cost per unit time of the long-running system.

2.2.2 Sensitivity

Consider a reward process $(X(t), \mathbf{r})$ where both the infinitesimal generator matrix $Q(\theta)$ and the reward rate vector $\mathbf{r}(\theta)$ may depend on some *parameters* $\theta \in \mathbb{R}^m$. The *sensitivity* analysis of the rewards $R(t)$ may reveal performance or reliability bottlenecks of the modeled system and aid designers in achieving desired performance measures.

Definition 2.6 The *sensitivity* of the expected reward rate $\mathbb{E}R(t)$ to the parameter $\theta[i]$ is the partial derivative

$$\frac{\partial\,\mathbb{E}R(t)}{\partial\,\theta[i]}.$$

The model reacts to the change of parameters with high absolute sensitivity more prominently, therefore they can be promising avenues of system optimization.

To calculate the sensivity of $\mathbb{E}R(t)$, the partial derivative of both sides of eq. (2.4) is taken, yielding

$$\frac{\partial \mathbb{E}R(t)}{\partial \theta[i]} = \frac{\partial \boldsymbol{\pi}(t)}{\partial \theta[i]}\mathbf{r}^{\mathrm{T}} + \boldsymbol{\pi}(t)\left(\frac{\partial \mathbf{r}}{\partial \theta[i]}\right)^{\mathrm{T}} = \mathbf{s}_i(t)\,\mathbf{r}^{\mathrm{T}} + \boldsymbol{\pi}(t)\left(\frac{\partial \mathbf{r}}{\partial \theta[i]}\right)^{\mathrm{T}},$$

where $\mathbf{s}_i$ is the sensitivity of $\boldsymbol{\pi}$ to the parameter $\theta[i]$.

In transient analysis, the sensitivity vector $\mathbf{s}_i$ is the solution of the initial value problem

$$\frac{\mathrm{d}\mathbf{s}_i(t)}{\mathrm{d}t} = \mathbf{s}_i(t)Q + \boldsymbol{\pi}(t)V_i, \quad \frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t} = \boldsymbol{\pi}_i(t)Q, \quad \mathbf{s}_i(0) = \mathbf{0}, \quad \boldsymbol{\pi}(0) = \boldsymbol{\pi}_0,$$

where $V_i = \partial Q(\theta)/\partial\theta[i]$ is the partial derivative of the generator matrix [38]. A similar initial value problem can be derived for the sensitivity of $\mathbf{L}(t)$ and $Y(t)$.

To obtain the sensitivity $\mathbf{s}_i$ of the steady-state probability vector $\boldsymbol{\pi}$, the system of linear equations

$$\mathbf{s}_i Q = -\boldsymbol{\pi}V_i, \quad \mathbf{s}_i\mathbf{1}^T = 0$$

is solved [3].

Another type of sensitivity analysis considers *unstructured* small perturbations of the infinitesimal generator matrix Q instead of dependecies on parameters [20, 25]. This latter, unstructured analysis may be used to study the numerical stability and conditioning of the solutions of the Markov chain.

2.2.3 Time to first failure

Let $D \subsetneq S$ be a set of *failure states* of the CTMC $X(t)$ and $U = S \setminus D$ be a set of operating states. We will assume without loss of generality that $U = \{0, 1, \ldots, n_U - 1\}$ and $D = \{n_U, n_U + 1, \ldots, n - 1\}$.

The matrix

$$Q_{UD} = \begin{pmatrix} Q_{UU} & \mathbf{q}_{UD}^{\mathrm{T}} \\ \mathbf{0} & 0 \end{pmatrix}$$

is the infinitesimal generator of a CTMC $X_{UD}(t)$ in which all the failures states D were merged into a single state n_U and all outgoing transitions from D were removed. The matrix Q_{UU} is the $n_U \times n_U$ upper left submatrix of Q, while the vector $\mathbf{q}_{UD} \in \mathbb{R}^{n_U}$ is defined as

$$q_{UD}[x] = \sum_{y \in D} q[x, y].$$

If the initial distribution $\boldsymbol{\pi}_0$ is 0 for all failure states (i.e. $\pi_0[x] = 0$ for all $x \in D$), the *Time to First Failure*

$$TFF = \inf\{t \geq 0 : X(t) \in D\} = \inf\{t \geq 0 : X_{UD}(t) = n_U\}$$

is *phase-type distributed* with parameters $(\boldsymbol{\pi}_U, Q_{UU})$ [34], where $\boldsymbol{\pi}_U$ is the vector containing the first n_U elements of $\boldsymbol{\pi}_0$. In particular, the *Mean Time to First Failure* is

$$MTFF = \mathbb{E}[TFF] = -\boldsymbol{\pi}_U Q_{UU}^{-1} \mathbf{1}^{\mathrm{T}}.$$

The probability of a D'-mode failure ($D' \in D$) is

$$\mathbb{P}(X(TFF_{+0}) = y) = -\boldsymbol{\pi}_D U Q_{UU}^{-1} \mathbf{q}_{UD'}^{\mathrm{T}},$$

where $\mathbf{q}_{UD'} \in \mathbb{R}^{n_U}$, $q_{UD'}[x] = \sum_{y \in D'} q[x, y]$ is the vector of transition rates from operational states to failure states D'.

2.3 Stochastic Petri nets

While reward processes based on continuous-time Markov chains allow the study of dependability or reliability measurements, the explicit specification of stochastic processes and rewards is often cumbersome. More expressive formalisms include queueing networks, stochastic process algebras such as PEPA [16, 21], Stochastic Automata Networks [19] and Stochastic Petri Nets (SPN).

Stochastic Petri Nets extend Petri nets by assigning random exponentially distributed random delays to transitions [29]. After the delay associated with an enabled transition is elapsed the transition fires *atomically* are transitions delays are reset.

Definition 2.7 A Stochastic Petri Net is a pair $SPN = (PN, \Lambda)$, where PN is a Petri net (P, T, F, W, M_0) and $\Lambda : T \to \mathbb{R}^+$ is a transition rate function.

Likewise, a stochastic Petri net with inhibitor arcs is a pair $SPN_I = (PN_I, \Lambda)$, where PN_I is a Petri net with inhibitor arcs.

A finite CTMC can be associated with a bounded stochastic Petri net (with inhibitor arcs) as follows:

1. The reachable state space of the Petri net is explored. We associate a consecutive natural numbers with the states such that the state space is

   $$RS = \{M_0, M_1, M_2, \ldots, M_{n-1}\},$$

   where M_0 is the initial marking. From now on, we will use markings $M_x \in RS$ and natural numbers $x \in \{0, 1, \ldots, n-1\}$ to refer to states of the model interchangably.

2. We define a CTMC $X(t)$ over the finite state space

   $$S = \{0, 1, 2, \ldots, n-1\}.$$

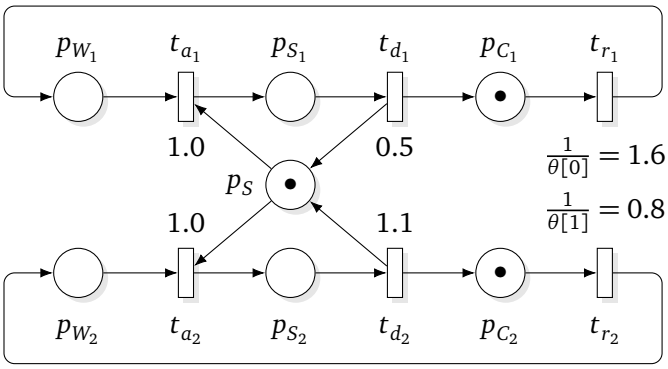


Figure 2.4 Example stochastic Petri net for the *SharedResource* model.

The initial distribution vector will be set to

$$\boldsymbol{\pi}(0) = \boldsymbol{\pi}_0 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

in the analysis steps ($\pi_0[x] = \delta_{0,x}$).

3. The generator matrix $Q \in \mathbb{R}^{n \times n}$ encodes the possible state transitions of the Petri net and the associated transition rate $\Lambda(\cdot)$ as

$$q_O[x, y] = \sum_{\substack{t \in T \\ M_x[t\rangle M_y}} \Lambda(t) \quad \text{if } x \neq y, \tag{2.5}$$

$$q_O[x, x] = 0,$$

$$Q = Q_O - \operatorname{diag}\{Q_O \mathbf{1}^{\mathrm{T}}\},$$

where the summation is done over all transition from the marking M_x to M_y, while Q_O and $Q_D = -\operatorname{diag}\{Q_O \mathbf{1}^{\mathrm{T}}\}$ are the off-diagonal and diagonal parts of Q, respectively.

Running example 2.5 Figure 2.4 shows the SPN model for *SharedResouce*, which is the Petri net from Figure 2.2 on page 5 extended with exponential transition rates.

The transitions a_1, d_1, a_2 and d_2 have rates 1.0, 0.5, 1.0 and 1.1, respectively. The parameter vector $\theta = (0.625, 1.25) \in \mathbb{R}^2$ is introduced such that the transitions r_1 and r_2 have rates $1/\theta[0]$ and $1/\theta[1]$.

The reachable state space (Table 2.1) contains 8 markings which are mapped to the integers $S = \{0, 1, \ldots, 7\}$. The state space graph along with the transition rates of

$RS =$

P:	S	C_1	W_1	S_1	C_2	W_2	S_2	
M_0	1	1	0	0	1	0	0	initial
M_1	1	0	1	0	1	0	0	client 1 waiting
M_2	1	1	0	0	0	1	0	client 2 waiting
M_3	1	0	1	0	0	1	0	1 waiting, 2 waiting
M_4	0	0	0	1	1	0	0	client 1 shared working
M_5	0	0	0	1	0	1	0	1 shared working, 2 waiting
M_6	0	1	0	0	0	0	1	client 2 shared working
M_7	0	0	1	0	0	0	1	1 waiting, 2 shared working

Table 2.1 Reachable state space of the *SharedResource* model.

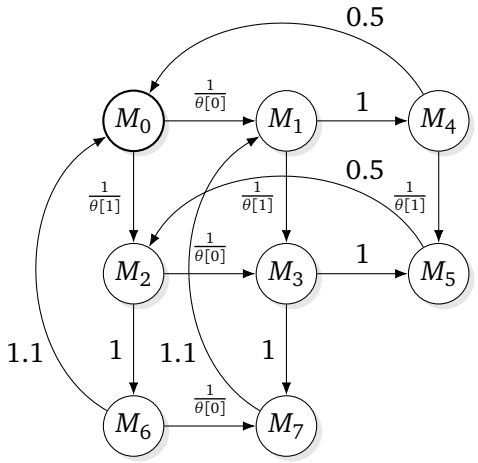


Figure 2.5 The CTMC associated with the *SharedResource* SPN model.

the CTMC is shown in Figure 2.5. The generator matrix is

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} * & \frac{1}{\theta[0]} & \frac{1}{\theta[1]} & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & \frac{1}{\theta[1]} & 1 & 0 & 0 & 0 \\ 0 & 0 & * & \frac{1}{\theta[0]} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & * & 0 & 1 & 0 & 1 \\ 0.5 & 0 & 0 & 0 & * & \frac{1}{\theta[1]} & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & * & 0 & 0 \\ 1.1 & 0 & 0 & 0 & 0 & 0 & * & \frac{1}{\theta[0]} \\ 0 & 1.1 & 0 & 0 & 0 & 0 & 0 & * \end{pmatrix} \end{array},$$

where in each row the diagonal element is the negative of the sum of the other elemens so that $Q\mathbf{1}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$. The CTMC is irreducible, therefore it has a well-defined steady-state distribution.

Extensions of stochastic Petri nets include transitions with general or phase-type delay distributions [28, 30], Generalized Stochastic Petri Nets (GSPN) with immediate transitions [31, 44] and Deterministic Stochastic Petri Nets (DSPN) with deterministic firing delays [41]. Among these, only phase-type distributed delays and GSPNs can be handled with purely Markovian analysis. Stochastic Well-formed Nets (SWN) are a class of colored Petri nets especially amenable to stochastic analysis [9]. Stochastic Activity Networks (SAN) also allow colored places, moreover, they introduce input and output gates for more flexible modeling [24].

2.3.1 Stochastic reward nets

Definition 2.8 A *Stochastic Reward Net* is a triple $SRN = (SPN, rr, ir)$, where SPN is a stochastic Petri net, $rr : \mathbb{N}^P \to \mathbb{R}$ is a *rate reward function* and $ir : T \times \mathbb{N}^P \to \mathbb{R}$ is an *impulse reward* function. A stochastic Reward net with inhibitor arcs is a triple $SRN_I = (SPN_I, rr, ir)$, where SPN_I is a stochastic Petri net with inhibitor arcs.

The rate reward $rr(M)$ is the reward gained per unit time in marking M, while $ir(t, M)$ is the reward gained when the transition t fires in marking M.

If $ir(t, M) \equiv 0$, the SRN is equivalent to the Markov reward process $(X(t), \mathbf{r})$, where $X(t)$ is the CTMC associated with the stochastic Petri net and

$$\mathbf{r} \in \mathbb{R}^n, \quad r[x] = rr(M_x).$$

If there are impulse rewards, exact calculation of the expected reward rate $\mathbb{E}R(t)$ and expected accumulated reward $\mathbb{E}Y(t)$ can be performed on reward process $(X, \mathbf{r})$,

$$r[x] = rr(M_x) + \sum_{t \in T, M_x[t\rangle} \Lambda(t)\, ir(t, M_x),$$

where the summation is taken over all enabled transitions [13]. In general, the distribution of $Y(t)$ cannot be derived by this method [37].

Running example 2.6 The SRN model

$$rr_1(M) = M(p_{S_1}) + M(p_{S_2}), \quad ir_1(t, M) \equiv 0$$

describes the utilization of the shared resouce in the *SharedResouce* SPN (Figure 2.4 on page 12). $R_1(t) = 1$ if the resource is allocated, hence $\mathbb{E}R_1(t)$ is the probability that the resource is in use at time t, while $Y(t)$ is the total usage time until t.

Another reward structure

$$rr_2(M) \equiv 0, \quad ir_2(t, M) = \begin{cases} 1 & \text{if } t \in \{t_{r_1}, t_{r_2}\}, \\ 0 & \text{otherwise} \end{cases}$$

counts the completed calculations, which are modeled by tokens leaving the places C_1 and C_2. The exprected steady-state reward rate $\lim_{t\to\infty} \mathbb{E}R(t)$ equals the number of calculations per unit time in a long-running system, while $Y(t)$ is the number of calculations performed until time t.

The reward vectors associated with these SRNs are

$$\begin{array}{lcccccccc} & 0 & & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \mathbf{r}_1 = \big(& 0 & & 0 & 0 & 0 & 1 & 1 & 1 & 1 \big), \\ \mathbf{r}_2 = \big(& \frac{1}{\theta[0]} + \frac{1}{\theta[1]} & & \frac{1}{\theta[1]} & \frac{1}{\theta[0]} & 0 & \frac{1}{\theta[1]} & 0 & \frac{1}{\theta[0]} & 0 \big). \end{array}$$

2.3.2 Superposed stochastic Petri nets

Definition 2.9 A *Superposed Stochastic Petri Net* (SSPN) is a pair $SSPN = (SPN, \mathcal{P})$, where $\mathcal{P} = \{P^{(0)}, P^{(1)}, \ldots, P^{(J-1)}\}$ is partition of the set of places $P = P^{(0)} \cup P^{(1)} \cup \cdots \cup P^{(J-1)}$ [15]. Superposed stochastic Petri nets with inhibitor arcs $SSPN_I = (SPN_I, \mathcal{P})$ are defined analogously.

The jth *local net* $LN^{(j)} = ((P^{(j)}, T^{(j)} = T_L^{(j)} \cup T_S^{(j)}, F^{(j)}, W^{(j)}, M_0^{(j)}), \Lambda^{(j)})$ can be constructed as follows:

- $P^{(j)}$ is the corresponding set from the partition of the original net.
- $T^{(j)}$ contains the local transition $T_L^{(j)}$ and synchronizing transitions $T_S^{(j)}$. A transition is *local* to $LN^{(j)}$ if it only affects places in $P^{(j)}$, that is,

$$T_L^{(j)} = \{t \in T : {}^\bullet t \cup t^\bullet \subseteq P^{(j)}\}. \tag{2.6}$$

  No transition may be local to more than one local net.

  A transition *synchronizes* with $LN^{(j)}$ if it affects some places in $P^{(j)}$ but it is not local to $LN^{(j)}$,

$$T_S^{(j)} = \{t \in T : ({}^\bullet t \cup t^\bullet) \cap P^{(j)} \neq \emptyset\} \setminus T_L^{(j)}. \tag{2.7}$$

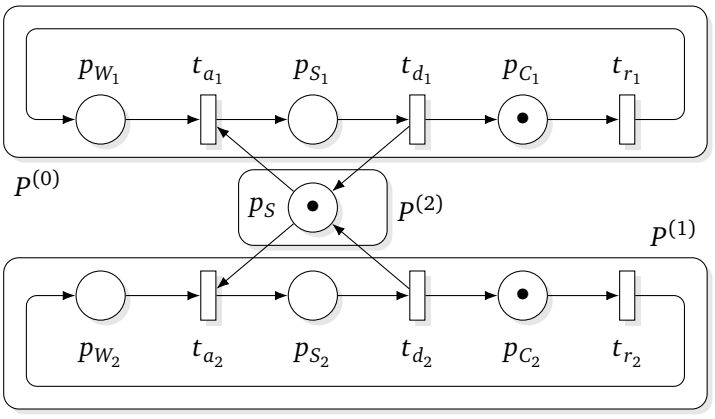


Figure 2.6 A partitioning of the *SharedResource* Petri net.

- The relation $F^{(j)}$ and the functions $W^{(j)}$, $M_0^{(j)}$, $\Lambda^{(j)}$ are the appropriate restrictions of the original structures, $F^{(j)} = F \cap ((P^{(j)} \times T^{(j)}) \cup (T^{(j)} \times J^{(j)}))$, $W^{(j)} = W|_{F^{(j)}}$, $M_0^{(j)} = M_0|_{P^{(j)}}$, $\Lambda^{(j)} = M_0|_{T^{(j)}}$.

If there are inhibitor arcs in $SSPN_I$, inhibitor arcs must be considered when local net $LN_I^{(j)}$ is constructed. The set ${}^\bullet t \cup t^\bullet$ is replaced with ${}^\bullet t \cup t^\bullet \cup {}^\circ t$ in eqs. (2.6) and (2.7) so that the enablement of local transitions only depends on the marking of places in $P^{(j)}$ and only places in $P^{(j)}$ may be affected upon firing. In addition, the inhibitor arc relation and weight function are restricted as $I^{(j)} = I \cap (P^{(j)} \cap T^{(j)})$, $W_I^{(j)} = W_I|_{I^{(j)}}$.

The set of all synchronizing transitions is denoted as $T_S = \bigcup_{j=0}^{J-1} T_S^{(j)}$. The *support* of the transition $t \in T$ is the set of components it is adjacent to, $\operatorname{supp} t = \{j : t \in T^{(j)}\}$.

Running example 2.7 Figure 2.6 shows a possible partitioning of the *SharedResource* SPN into a SSPN. The components $P^{(0)}$ and $P^{(1)}$ model the two consumers, while $P^{(2)}$ contains the unallocated resource S.

The transitions r_1 and r_2 are local to $LN^{(0)}$ and $LN^{(1)}$, respectively, while a_1, d_1, a_2 and d_2 synchronize with $LN^{(2)}$ and the local net associated with their consumers.

The *local reachable state space* $RS^{(j)}$ of $LN^{(j)}$ is the the set of markings beloning to the state space RS of the original net restricted to the places $P^{(j)}$ (duplicates removed),

$$RS^{(j)} = \{M^{(j)} : M \in RS, M^{(j)} = M|_{P^{(j)}}\}.$$

This is a *subset* of the reachable state space of $LN^{(j)}$, in particular, $RS^{(j)}$ is always finite if RS is finite, even if $LN^{(j)}$ is not bounded. Analysis techniques for generating local

$$RS^{(0)} = \left\{ \begin{array}{|c|ccc|} \hline P: & C_1 & W_1 & S_1 \\ \hline M_0^{(0)} & 1 & 0 & 0 \\ M_1^{(0)} & 0 & 1 & 0 \\ M_2^{(0)} & 0 & 0 & 1 \\ \hline \end{array} \right\},$$

$$RS^{(1)} = \left\{ \begin{array}{|c|ccc|} \hline P: & C_2 & W_2 & S_2 \\ \hline M_0^{(1)} & 1 & 0 & 0 \\ M_1^{(1)} & 0 & 1 & 0 \\ M_2^{(1)} & 0 & 0 & 1 \\ \hline \end{array} \right\}, \quad RS^{(2)} = \left\{ \begin{array}{|c|c|} \hline P: & S \\ \hline M_0^{(2)} & 1 \\ M_1^{(2)} & 0 \\ \hline \end{array} \right\}$$

Table 2.2 Local reachable markings of the *SharedResouce* SSPN from Figure 2.6.

state spaces include *partial P-invariants* [8] and explicit projection of global reachable markings [5].

The *potential state space PS* of an SSPN is the Descares product of the local reachable state spaces of its components

$$PS = RS^{(0)} \times RS^{(1)} \times \cdots \times RS^{(J-1)},$$

which is a (possibly not proper) superset of the global reachable state space RS.

We will associate the natural numbers $S^{(j)} = \{0, 1, \ldots, n_j - 1\}$ with the local reachable markings $RS^{(j)} = \{M_0, M_1, \ldots, M_{n_j-1}\}$ to aid the construction of Markov chains and use them interchangably. The notation

$$M = \mathbf{x} = (x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}) \tag{2.8}$$

refers to the global state $\mathbf{x}$ composed from the local markings $x^{(j)}$, i.e. the marking

$$M(p) = M^{(j)}_{x^{(j)}}(p), \quad \text{if } p \in P^{(j)},$$

which is the union of the local markings $M^{(0)}_{x^{(0)}}, M^{(1)}_{x^{(1)}}, \ldots, M^{(J-1)}_{x^{(J-1)}}$.

Running example 2.8 The local reachable markings of the *SharedResource* SSPN are enumerated in Table 2.2.

The transitions d_1 and d_2 are always enabled in $LN^{(2)}$ because all their input places are located in other components, thus $LN^{(2)}$ is an unbounded Petri net. Despite this, $RS^{(2)}$ is finite, because it only contains the local markings which are reachable in the original net.

The potential state space PS contains $3 \cdot 3 \cdot 2 = 18$ potential markings, although only 8 are reachable (Table 2.1 on page 13). For example, the marking $(2, 2, 0)$ is not reachable, as it would violate mutual exclusion.

2.4 Kronecker algebra

Definition 2.10 The *Kronecker product* of matrices $A \in \mathbb{R}^{n_1 \times m_1}$ and $B \in \mathbb{R}^{n_2 \times m_2}$ is the matrix $C = A \otimes B \in \mathbb{R}^{n_1 n_2 \times m_1 m_3}$, where

$$c[i_1 n_1 + i_2, j_1 m_1 + j_2] = a[i_1, j_1] b[i_2, j_2].$$

Some properties of the Kroncker product are

1. Associativity:
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C,$$
which makes J-way Kronecker products $A^{(0)} \otimes A^{(1)} \otimes \cdots \otimes A^{(J-1)}$ well-defined.
2. Distributivity over matrix addition:
$$(A + B) \otimes (C + D) = A \otimes C + B \otimes C + A \otimes D + B \otimes D,$$
3. Compatibility with ordinary matrix multiplication:
$$(AB) \otimes (CD) = (A \otimes C)(B \otimes D),$$
in particular,
$$A \otimes B = (A \otimes I_2)(I_1 \otimes B)$$
for appropriately-sized identity matrices I_1 and I_2.

We will occasionally employ multi-index notation to refer to elements of Kronecker product matrices. For example, we will write

$$b[\mathbf{x}, \mathbf{y}] = b[(x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}), (y^{(0)}, y^{(1)}, \ldots, y^{(J-1)})] = a^{(0)}[x^{(0)}, y^{(0)}] a^{(1)}[x^{(1)}, y^{(1)}] \cdots a^{(J-1)}[x^{(J-1)}, y^{(J-1)}],$$

where $\mathbf{x} = (x^{(0)}, x^{(1)}, \ldots, x^{(J-1)})$, $\mathbf{y} = (y^{(0)}, y^{(1)}, \ldots, y^{(J-1)})$ and B is the J-way Kronecker product $A^{(0)} \otimes A^{(1)} \otimes \cdots \otimes A^{(J-1)}$.

Definition 2.11 The *Kronecker sum* of matrices $A \in \mathbb{R}^{n_1 \times m_1}$ and $B \in \mathbb{R}^{n_2 \times m_2}$ is the matrix $C = A \oplus B \in \mathbb{R}^{n_1 n_2 \times m_1 m_3}$, where

$$C = A \otimes I_2 + I_1 \otimes B,$$

where $I_1 \in \mathbb{R}^{n_1 \times m_1}$ and $I_2 \in \mathbb{R}^{n_2 \times m_2}$ are identity matrices.

Example 2.9 Consider the matrices

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}.$$

Their Kronecker product is

$$A \otimes B = \begin{pmatrix} 1\cdot 0 & 1\cdot 1 & 2\cdot 0 & 2\cdot 1 \\ 1\cdot 2 & 1\cdot 0 & 2\cdot 2 & 2\cdot 0 \\ 3\cdot 0 & 3\cdot 1 & 4\cdot 0 & 4\cdot 1 \\ 3\cdot 2 & 3\cdot 0 & 4\cdot 2 & 4\cdot 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 2 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \\ 6 & 0 & 8 & 0 \end{pmatrix},$$

while their Kronecker sum is

$$A \oplus B = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 0 \\ 2 & 1 & 0 & 2 \\ 3 & 0 & 4 & 1 \\ 0 & 3 & 2 & 4 \end{pmatrix}.$$

Chapter 4

Efficient generation and storage of continuous-time Markov chains

4.1 Explicit methods

4.1.1 Explicit state space and matrix construction

Explicit state space enumeration for Petri nets repeatedly applies the firing rule eq. (2.1) on page 4 starting from the initial marking M_0 until no new marking can be generated. At the end of the enumeration of the finite state space, all reachable markings $M_0 \rightsquigarrow M$ are discovered. We implemented detection of already encountered markings by hashing, while new markings are generated by breath-first search.

Given the finite state space of size $n = |RS|$ in an explicit form along with a bijection between the markings an the natural numbers $\{0, 1, \ldots, n-1\}$, the generator matrix Q can be directly created by Algorithm 4.1. The algorithm stores the transition rate $\Lambda(t)$ in Q for all pairs of reachable markings $M_x\,[t\rangle\,M_y$ and transitions $t \in T$.

The generator matrix requires $O(n^2)$ memory if a two-dimensional dense array format is used. Because firing a transition can only take the Petri net from a given marking M_x to a single target marking M_y in the SPN formalism, each column of Q may contain up to $|T|$ nonzero elements. Hence Q requires $O(|T|n)$ memory if a sparse format is chosen.

Unfortunately, both of these storage methods may be prohibitively costly for large models due to state space explosion. In addition, explicit enumeration of large RS may take an extreme amout of time.

Algorithm 4.1 Generator matrix construction from explicit state space.

Input: explicit state space RS, transitions T, transition rate function Λ
Output: generator matrix Q

```
allocate Q_O ∈ ℝ^{|RS|×|RS|}, d ∈ ℝ^{|RS|}
foreach y ∈ RS, t ∈ R do
    if there is a state x ∈ RS such that M_x [t⟩ M_y then
        q_D[x, y] ← q_D[x, y] + Λ(t)
d ← −Q_O 1^T
return Q_O + diag{d}
```

4.1.2 Block Kronecker generator matrices

Kronecker generator matrices

To alleviate the high memory requirements of Q, the Kronecker decomposition for a superposed SPN with J components expresses the infinitesimal generator matrix of the associated CTMC in the form

$$Q = Q_O + Q_D, \quad Q_O = \bigoplus_{j=0}^{J-1} Q_L^{(j)} + \sum_{t \in T_S} \Lambda(t) \bigotimes_{j=0}^{J-1} Q_t^{(j)}, \quad Q_D = -\operatorname{diag}\{Q_O \mathbf{1}^{\mathrm{T}}\}, \tag{4.1}$$

where Q_O and Q_D are the off-diagonal and diagonal parts of Q. The matrix

$$Q_L^{(j)} = \sum_{t \in T_L^{(j)}} \Lambda(t) Q_L^{(j)}$$

is the *local* transition matrix of the component j, while the matrix

$$Q_t^{(j)} \in \mathbb{R}^{n_j \times n_j}, \quad q_t^{(j)}[x^{(j)}, y^{(j)}] = \begin{cases} 1 & \text{if } x^{(j)}\,[t\rangle\, y^{(j)}, \\ 0 & \text{otherwise} \end{cases}$$

describes the effects of the transition t on $LN^{(j)}$. $Q_t^{(j)}$ has a nonzero element for every local state transition caused by t. If $j \notin \operatorname{supp} t$, $Q_t^{(j)}$ is an $n_j \times n_j$ identity matrix.

It can be seen that

$$\begin{aligned} q_O[\mathbf{x}, \mathbf{y}] &= \sum_{j=0}^{J-1} \sum_{t \in T_L^{(j)}} \Lambda(t)\, q_t^{(j)}[x^{(j)}, y^{(j)}] + \sum_{t \in T_S} \Lambda(t) \prod_{j=0}^{J-1} q_t^{(j)}[x^{(j)}, y^{(j)}] \\ &= \sum_{j=0}^{J-1} \sum_{\substack{t \in T_L^{(j)} \\ x^{(j)}[t\rangle y^{(j)}}} \Lambda(t) + \sum_{\in T_S, \mathbf{x}[t\rangle \mathbf{y}} \Lambda(t) = \sum_{t \in T, \mathbf{x}[t\rangle \mathbf{y}} \Lambda(t), \end{aligned} \tag{4.2}$$

which is the same as eq. (2.5) on page 12. Indeed, eq. (4.1) is a representation of the infinitesimal generator matrix.

The matrices $Q_L^{(j)}$ and $Q_t^{(j)}$ and the vector $-Q_O\mathbf{1}^{\mathrm{T}}$ together are usually much smaller than the full generator matric Q even when stored in a sparse matrix form. Hence Kronecker decomposition may save a significant amount of storage at the expense of some computation time.

Unfortunately, the Kronecker generator Q is a $n_0n_1\cdots n_{J-1} \times n_0n_1\cdots n_{J-1}$ matrix, i.e. in encodes the state transitions in the potential state space PS instead of the reachable state space RS.

Potential Kronecker methods [6] perform computations with the $|PS| \times |PS|$ Q matrix and vectors of length $|PS|$. In addition to increasing storage requirements, this may lead to problems in some numerical solution algorithms, because the CTMC over PS is not neccessarily irreducible even if it is irreducible over RS.

In contrast, *actual Kronecker methods* [2, 6, 26] work with vectors of length $|RS|$. However, additional conversions must be performed between the actual dense indexing of the vectors and the potential sparse indexing of the Q matrix, which leads to implementation complexities and computational overhead.

A third approach, which we discuss in the next subsection, imposes a hierarchical structure on RS [1, 5, 8].

Macro state construction

The hierarchical structuring of the reachable state space expresses RS as

$$RS = \bigcup_{\tilde{\mathbf{x}}\in\widetilde{RS}} \prod_{j=0}^{J-1} RS^{(j)}_{\tilde{x}^{(j)}}, \quad RS^{(j)} = \bigcup_{\tilde{x}^{(j)}\in\widetilde{RS}^{(j)}} RS^{(j)}_{\tilde{x}^{(j)}},$$

where $\widetilde{RS} = \{\tilde{0}, \tilde{1}_1, \ldots, \widetilde{\tilde{n}-1}\}$ a set of *global macro states*, $\widetilde{RS}^{(j)} = \{\tilde{0}^{(j)}, \tilde{1}^{(j)}, \ldots, \widetilde{\tilde{n}_j-1}^{(j)}\}$ is the set of *local macro states* of $LN^{(j)}$, and $RS_x^{(j)} = \{0_x^{(j)}, 1_x^{(j)}, \ldots, (n_{j,x}-1)_x^{(j)}\}$ are the *local micro states* in the local macro state $\tilde{x}^{(j)}$. The product symbol denotes the composition of local markings, as in eq. (2.8) on page 17.

The local micro states form a partition $RS^{(j)} = \bigcup_{x\in\widetilde{RS}^{(j)}} RS_x^{(j)}$ of the state space of the jth SSPN component.

Construction of macro states is performed as follows [5]:

1. The equivalence relation $\sim^{(j)}$ is defined over $RS^{(j)}$ as

$$x^{(j)} \sim^{(j)} y^{(j)} \iff \{\hat{\mathbf{z}}^{(j)} : \mathbf{x}\in RS, z^{(j)} = x^{(j)}\} = \{\hat{\mathbf{z}}^{(j)} : \mathbf{x}\in RS, z^{(j)} = y^{(j)}\}, \tag{4.3}$$

   where $\hat{\mathbf{z}}^{(j)} = (z^{(0)}, \ldots, z^{(j-1)}, z^{(j+1)}, \ldots, z^{(J-1)})$, i.e. two local states are equivalent if they are reachable in the same combinations of ambient local markings.

Therefore, the relation

$$\mathbf{x} \sim \mathbf{y} \quad \Longleftrightarrow \quad x^{(j)} \sim^{(j)} y^{(j)} \text{ for all } j = 0, 1, \ldots, J-1,$$

defined over PS, has the property that whether $\mathbf{x} \sim \mathbf{y}$, either both $\mathbf{x}$ and $\mathbf{y}$ are reachable (global) markings, or neither are.

2. Reachable local macro states are the partitions of $RS^{(j)}$ generated by $\sim^{(j)}$. A bijection $\widetilde{RS}^{(j)} \leftrightarrow RS^{(j)}/\sim^{(j)}$ is formed between the integers $0, 1, \ldots, \tilde{n}^{(j)} - 1$ and the local state partitions for each component $LN^{(j)}$.

3. The set of potential macro states is

$$\widetilde{PS} = \prod_{j=0}^{J-1} \widetilde{RS}^{(j)} \supseteq \widetilde{RS}$$

the Descares product of the local macro states. If macro state $\tilde{x} \in \widetilde{PS}$ contains a reachable state, all associated (micro) states are reachable, because $\widetilde{PS}$ is the partition $RS/\sim$ of RS generated by the relation $\sim$. Thus, $\widetilde{RS}$ is constructed by enumerating the reachable macro states in $\widetilde{PS}$. A bijection is formed between the reachable subset of $\widetilde{PS}$ and the integers $\tilde{0}, \tilde{1}, \ldots, \widetilde{\tilde{n}-1}$.

The pseudocode for this process is shown in Algorithm 4.2. The decomposition is extremely memory intensive due to the allocation of the bit vector $\mathbf{b}$ of length $|PS|$.

In [5], sorting the columns of B lexicographically was recommended to calculate the partition $B/=$ In our implementation, we insert the rows of B into a bitwise trie and detect duplicates instead, so that no mapping between the original order and sorted ordering of columns needs to be maintained.

Running example 4.1 The macro states of the *RunningExample* SSPN model (Figure 2.6 on page 16) are obtained from its component state space (Table 2.2 on page 17) as follows:

1. The bit vector $\mathbf{b}$ is filled according to the reachable states RS,

$$\mathbf{b} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix},$$

where the mixed indices in small type refer to the states of the local nets $LN^{(0)}$, $LN^{(1)}$ and $LN^{(2)}$.

Algorithm 4.2 Hiearchical decomposition of the reachable state space into macro states by Buchholz [5].

Input: Reachable state space RS, reachable local state spaces $RS^{(j)}$
Output: Macro state space $\widetilde{RS}$, local macro state spaces $\widetilde{RS}^{(j)}$

allocate bit vector $\mathbf{b} \in \{0,1\}^{n_0 n_1 \cdots n_{J-1}}$ initialized with zeroes
foreach $\mathbf{x} \in RS$ **do**
  // Fill $\mathbf{b}$ with ones corresponding to reachable states
  $b[n_{J-1}n_{J-2}\cdots n_1 x^{(0)} + n_{J-1}n_{J-2}\cdots n_2 x^{(1)} + \cdots + n_{J-1}x^{(J-2)} + x^{(J-1)}] \leftarrow 1$
for $j \leftarrow 0$ **to** $J-1$ **do**
  Reshape $\mathbf{b}$ into matrix B with n_j columns
  Partition the columns of B by componentwise equality
  foreach subset S of the partition $B/=$ **do**
    Create a new local macro state $\tilde{y}^{(j)}$ in $\widetilde{RS}^{(j)}$
    Assign all local micro states $z \in S$ to $\tilde{y}^{(j)}$
    Drop all columns of B corresponding to S but a single representant of $\tilde{y}^{(j)}$
foreach $\tilde{\mathbf{x}} \in RS^{(0)} \times RS^{(1)} \times \cdots \times RS^{(J-1)}$ **do**
  if $b[\tilde{\mathbf{x}}] = 1$ **then** Add $\tilde{\mathbf{x}}$ to $\widetilde{RS}$ as a global macro state
return $\widetilde{RS}, \{\widetilde{RS}^{(j)}\}_{j=0}^{J-1}$

2. We reshape $\mathbf{b}$ into a matrix B so that each column corresponds to a local state of the component $LN^{(0)}$,

$$B = \begin{array}{c} \\ 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \\ 2\,0 \\ 2\,1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

in order to conclude that

$$\widetilde{RS}_0^{(0)} = \{0_0^{(0)} = M_0^{(0)}, 1_0^{(0)} = M_1^{(0)}\}, \qquad \widetilde{RS}_1^{(0)} = \{0_1^{(0)} = M_2^{(0)}\}.$$

3. After removing all local states of $LN^{(0)}$ except representants of $\widetilde{RS}^{(0)}$, $\mathbf{b}$ is

reshaped again

$$B = \begin{array}{c} \\ \tilde{0}\,0 \\ \tilde{0}\,1 \\ \tilde{1}\,0 \\ \tilde{1}\,1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

to find that

$$\widetilde{RS}_0^{(1)} = \{0_0^{(1)} = M_0^{(1)}, 1_0^{(1)} = M_1^{(1)}\}, \qquad \widetilde{RS}_1^{(0)} = \{0_1^{(1)} = M_2^{(1)}\}.$$

4. Finally, we reshape

$$B = \begin{array}{c} \\ \tilde{0}\,\tilde{0} \\ \tilde{0}\,\tilde{1} \\ \tilde{1}\,\tilde{0} \\ \tilde{1}\,\tilde{1} \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \end{array}$$

and conclude

$$\widetilde{RS}_0^{(2)} = \{0_0^{(2)} = M_0^{(2)}\}, \qquad \widetilde{RS}_1^{(2)} = \{0_1^{(2)} = M_1^{(2)}\}.$$

5. Unfolding the matrix B

$$\mathbf{b} = \begin{array}{c} \begin{array}{cccccccc} \tilde{0} & \tilde{0} & \tilde{0} & \tilde{0} & \tilde{1} & \tilde{1} & \tilde{1} & \tilde{1} \end{array} \\ \begin{array}{cccccccc} \tilde{0} & \tilde{0} & \tilde{1} & \tilde{1} & \tilde{0} & \tilde{0} & \tilde{1} & \tilde{1} \end{array} \\ \begin{array}{cccccccc} \tilde{0} & \tilde{1} & \tilde{0} & \tilde{1} & \tilde{0} & \tilde{1} & \tilde{0} & \tilde{1} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

shows that the reachable global macro states are

$$\widetilde{RS} = \{\tilde{0} = (\tilde{0}^{(0)}, \tilde{0}^{(1)}, \tilde{0}^{(2)}), \tilde{1} = (\tilde{0}^{(0)}, \tilde{1}^{(1)}, \tilde{1}^{(2)}), \tilde{2} = (\tilde{1}^{(0)}, \tilde{0}^{(1)}, \tilde{1}^{(2)})\},$$

where $\tilde{0}$ corresponds to the free state of the resource, while in $\tilde{1}$ and $\tilde{2}$, the clients $LN^{(1)}$ and $LN^{(0)}$ are using the resource, respectively.

Block kronecker matrix composition

The *hierarchical* or *block* Kronecker form of Q expresses the infinitesimal generator of the CTMC over the reachable state space by the means of macro state decomposition.

The matrices $Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}]$ and $Q_L^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] \in \mathbb{R}^{n_{j,x} \times n_{n,y}}$ describe the effects of a

single transition $t \in T$ and the aggregate effects of local transitions on $LN^{(j)}$ as its state changes from the local macro state $\tilde{x}^{(j)}$ to $\tilde{y}^{(j)}$, respectively. Formally,

$$q_t^{(j)}[\tilde{x}^{(j)}, \tilde{y}^{(j)}][a_x^{(j)}, b_y^{(j)}] = \begin{cases} 1 & \text{if } a_x^{(j)}\,[t\rangle\, b_y^{(j)}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.4}$$

$$Q_L^{(j)}[\tilde{x}^{(j)}, \tilde{y}^{(j)}] = \sum_{t \in T_L^{(j)}} \Lambda(t)\, Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{y}^{(j)}]. \tag{4.5}$$

In the case $j \notin \operatorname{supp} t$, we define $Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{y}^{(j)}]$ as an identity matrix if $\tilde{x}^{(j)} = \tilde{y}^{(j)}$ and a zero matrix otherwise.

Let us call macro state pairs $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *single local macro state transitions* (slmst.) at h if $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ differ only in a single index h ($\tilde{x}^{(h)} \neq \tilde{y}^{(j)}$).

The off-diagonal part Q_D of Q is written as a block matrix with $\tilde{n} \times \tilde{n}$ blocks. A single block is expressed as

$$Q_O[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}] = \begin{cases} \displaystyle\bigoplus_{j=0}^{J-1} Q_L^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] \\ \qquad + \displaystyle\sum_{t \in T_S} \Lambda(t) \bigotimes_{j=0}^{J-1} Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] & \text{if } \tilde{\mathbf{x}} = \tilde{\mathbf{y}}, \\ I_{N_1 \times N_1} \otimes Q_L^{(h)}[\tilde{x}^{(h)}, \tilde{x}^{(h)}] \otimes I_{N_1 2 \times N_2} \\ \qquad + \displaystyle\sum_{t \in T_S} \Lambda(t) \bigotimes_{j=0}^{J-1} Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] & \text{if } (\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \text{ is slmst. at } h, \\ \displaystyle\sum_{t \in T_S} \Lambda(t) \bigotimes_{j=0}^{J-1} Q_t^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] & \text{otherwise,} \end{cases}$$

where $I_1 = \prod_{f=0}^{h-1} n_{h,x^{(h)}}$, $I_2 = \prod_{f=h+1}^{J-1} n_{h,x^{(h)}}$. If $\mathbf{x} = \mathbf{y}$, the matrix block describes transition which leave the global macro state unchanged, therefore any local transition may fire. If $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is slmst. at h, only local transitions on the component h may cause the global state transition, since no other local transition may affect $LN^{(h)}$. In every other case, only synchronizing transitions may occur.

This expansion of block matrices is equivalent to eq. (4.1) on page 30 except the considerations to the hierarchical structure of the state space.

The full Q matrix is written as

$$Q = Q_O + Q_D, \quad Q_D = -\operatorname{diag}\{Q_O \mathbf{1}^{\mathrm{T}}\}$$

as usual.

Algorithm 4.3 shows the construction of the local transition matrices according to eqs. (4.4) and (4.5).

Algorithm 4.3 Transition matrix construction for block Kronecker matrices

Input: State spaces $\widetilde{RS}^{(j)} RS_x^{(j)}$, transitions T, transition rates Λ
Output: Transition matrices $Q_t^{(j)}, Q_L^{(j)}$

```
1  for j ← 0 to J − 1 do
2      foreach (x̃^(j), ỹ^(j)) ∈ R̃S^(j) × R̃S^(j) do
3          if j ∈ supp t then
4              allocate Q_t^(j)[x̃^(j), ỹ^(j)] ∈ ℝ^{n_{j,x} × n_{n,y}}
5              Fill in Q_t^(j)[x̃^(j), ỹ^(j)] according to eq. (4.4) on page 35
6          else if x̃^(j) = ỹ^(j) then Q_t^(j)[x̃^(j), ỹ^(j)] ← I_{n_{j,x} × n_{j,y}}
7          else Q_t^(j)[x̃^(j), ỹ^(j)] ← 0_{n_{j,x} × n_{j,y}}
8      allocate Q_L^(j)[x̃^(j), ỹ^(j)] ∈ ℝ^{n_{j,x} × n_{n,y}}
9      foreach t ∈ T_L^(j) do
10         Q_L^(j)[x̃^(j), ỹ^(j)] ← Q_L^(j)[x̃^(j), ỹ^(j)] + Λ(t) Q_t^(j)[x̃^(j), ỹ^(j)]
```

The construction of the block matrix Q is shown in Algorithm 4.4 on page 44. The formulation from eq. (4.2) is optimized in several ways:

- If a Kronecker product contains a 0 matrix term, it is itself zero, therefore, such products are discarded in line 23.
- For identity matrices $I_{N\times N} \otimes I_{n\times n}$ holds. This is exploited in line 21 to reduce the number of terms in the Kronecker producs.
- Instead of constructing Q_O and Q_D separately, the diagonal elements are added to the blocks of Q along its diagonal in line 26.

4.2 Symbolic methods

4.2.1 Multivalued decision diagrams

Multivalued decision diagrams (MDDs) [12] provide a compact, graph-based representation for functions of the form $\mathbb{N}^J \to \{0, 1\}$.

Definition 4.1 A quasi-reduced ordered *multivalued decision diagram* (MDD) encoding the function $f(x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}) \in \{0, 1\}$ (where the domain of each variable $x^{(j)}$ is $D^{(j)} = \{0, 1, \ldots, n_j - 1\}$) is a tuple $MDD = (V, \underline{r}, \underline{0}, \underline{1}, level, children)$, where

- $V = \bigcup_{i=0}^{J} V_i$ is a finite set of *nodes*, where $V_0 = \{\underline{0}, \underline{1}\}$ are the *terminal nodes*, the rest of the nodes $V_N = V \setminus V_0$ are *nonterminal nodes*;
- $level : V \to \{0, 1, \ldots, J\}$ assigns nonnegative *level numbers* to each node ($V_i = \{\underline{v} \in V : level(\underline{v}) = i\}$);
- $\underline{v} \in V_J$ is the *root node*;
- $\underline{0}, \underline{1} \in V_0$ are the *zero* and *one terminal nodes*;
- $children : \left(\bigcup_{i=1}^{J} V_i \times D^{(i-1)}\right) \to V$ is a function defining edges between nodes labeled by the items of the domains, such that either $children(\underline{v}, x) = \underline{0}$ or $level(children(\underline{v}, x)) = level(\underline{v}) - 1$ for all $\underline{v} \in V$, $x \in D^{(level(\underline{v})-1)}$,
- if $\underline{n}, \underline{m} \in V_j, j > 0$ then the subgraphs formed by the nodes reachable from $\underline{n}$ and $\underline{m}$ are either non-isomorphic, or $\underline{n} = \underline{m}$.

We remark that due to the presence of the terminal level V_0 the indexing of the levels and the domains is shifted, i.e. the level V_i corresponds to the domain $D^{(i-1)}$.

According to the semantics of MDDs, $f(\mathbf{x}) = 1$ if the node $\underline{1}$ is reachable from $\underline{r}$ through the edges labeled with $x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}$,

$$f(x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}) = 1 \iff children(children(\ldots children(\underline{r}, x^{(J-1)}) \ldots, x^{(1)}), x^{(0)}) = \underline{1}.$$

Definition 4.2 A quasi-reduced ordered *edge-valued multivalued decision diagram* (EDD) [40] encoding the function $g(x^{(0)}, x^{(1)}, \ldots, x^{(J-1)}) \in \mathbb{N}$ is a tuple $EDD = (V, \underline{r}, \underline{0}, \underline{1}, level, children, label)$, where

- $MDD = (V, \underline{r}, \underline{0}, \underline{1}, level, children)$ is a quasi-reduced ordered MDD,
- $label : \left(\bigcup_{i=1}^{J} V_i \times D^{(i-1)}\right) \to \mathbb{N}$ is an edge label function.

According to the semantics of EDDs, the function g is evaluated as

$$g(\mathbf{x}) = \begin{cases} \text{undefined} & \text{if } f(\mathbf{x}) = 0, \\ \sum_{j=0}^{J-1} label(\underline{n}^{(j)}, x^{(j)}) & \text{if } f(\mathbf{x}) = 1, \end{cases}$$

where f is the function associated with the underlying MDD and $\underline{n}^{(j)}$ are the nodes along the path to $\underline{1}$, i.e.

$$\underline{n}^{(J-1)} = \underline{r}, \quad \underline{n}^{(j)} = children(\underline{n}^{(j+1)}, x^{(j+1)}).$$

4.2.2 Symbolic state spaces

Symbolic techniques involving MDDs can efficiently store large reachable state spaces of superposed Petri nets. Reachable states $x \in RS$ are associated with state codings

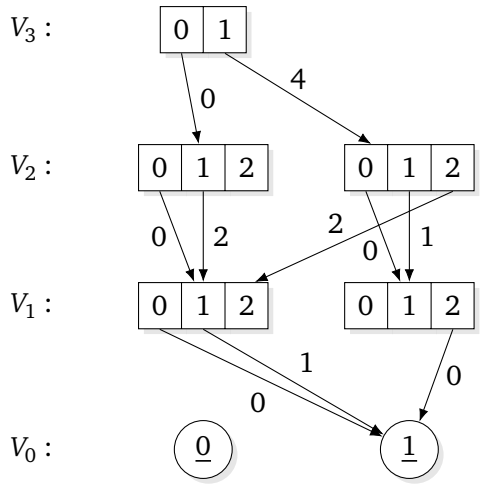


Figure 4.1 EDD state space mapping for the *SharedResource* SSPN.

$\mathbf{x} = (x^{(0)}, x^{(1)}, \ldots, x^{(J-1)})$. The function $f : PS \to \{0, 1\}$ can be stored as an MDD where $f(\mathbf{x}) = 1$ if and only if $x \in RS$. The domains of the MDD are the local state spaces $D^{(j)} = RS^{(j)}$.

Similarly, EDDs can efficiently store the mapping between symbolic state encodings $\mathbf{x}$ and reachable state indices $x \in RS = \{0, 1, \ldots, n-1\}$ as the function $g(\mathbf{x}) = x$. This mapping is used to refer to elements of state probability vectors $\boldsymbol{\pi}$ and the sparse generator matrix Q when these objects are created and accessed.

Running example 4.2 Figure 4.1 shows the state space of the *SharedResource* model encoded as an EDD. The edge labels express the lexicographic mapping of symbolic state codes $\mathbf{x}$ to state indices $\mathbf{x}$. Edges to the terminal zero node $\underline{0}$ were omitted for the sake of clarity.

Some iteration strategies for MDD state space exploration are *breath-first search* and *saturation* [12]. We use the implementation of saturation from the Petridotnet framework [14, 18].

Algorithms 4.5 and 4.6 on page 45 illustrate the construction of a generator matrix based on the state space encoded as EDDs. The procedure FillIn descends the EDD following a path for the target and the source state simulatenously. The edge labels, represeting state indices, are summed on both paths. If a transition $\mathbf{x}\,[t\rangle\,\mathbf{y}$ is found, the matrix element $q_O[x, y]$ corresponding to the summed indices in increated by the transition rate $\Lambda(t)$. Algorithm 4.6 repeats FillIn for all transitions.

4.2.3 Symbolic hierarchical state space decomposition

The memory requirements and runtime of Algorithm 4.2 on page 33 may be significantly improved by the use of symbolic state space storage instead of a bit vector.

To symbolically partition the local states $RS^{(j)}$ into macro states $\widetilde{RS}^{(j)}$, we will use the following notations of above and below substates from Ciardo et al. [11]:

Definition 4.3 The set of *above* substates coded by the node $\underline{n}$ is

$$\mathcal{A}(\underline{n}) \subseteq \{(x^{(j+1)}, x^{(j+2)}, \ldots, x^{(J-1)}) \in RS^{(j+1)} \times RS^{(j+2)} \times \cdots \times RS^{(J-1)}\},$$

such that

$$\mathbf{x} \in \mathcal{A}(\underline{n}) \iff children(children(\ldots children(\underline{r}, x^{(J-1)}) \ldots, x^{(j+2)}), x^{(j+1)}) = \underline{n}$$

and $j = level(\underline{n}) - 1$, i.e. $\mathcal{A}(\underline{n})$ is the set of all paths in the MDD leading from $\underline{r}$ to $\underline{n}$.

Definition 4.4 The set of *below* substates coded by the node $\underline{n}$ is

$$\mathcal{B}(\underline{n}) \subseteq \{(x^{(0)}, x^{(1)}, \ldots, x^{(j)}) \in RS^{(0)} \times RS^{(1)} \times \cdots \times RS^{(j)}\},$$

such that

$$\mathbf{x} \in \mathcal{B}(\underline{n}) \iff children(children(\ldots children(\underline{n}, x^{(j)}) \ldots, x^{(1)}), x^{(0)}) = \underline{1}$$

and $j = level(\underline{n}) - 1$, i.e. $\mathcal{B}(\underline{n})$ is the set of all paths in the MDD leading from $\underline{n}$ to $\underline{1}$.

The relation $\sim^{(j)}$ over $RS^{(j)}$ can be expressed with $\mathcal{A}(\underline{n})$ and $\mathcal{B}(\underline{n})$ is a way that can be handled easily with symbolic techniques.

Observation 4.5 The set of states which contain some local state $x^{(j)}$ is

$$\{\hat{\mathbf{z}}^{(j)} : \mathbf{z} \in RS, z^{(j)} = x^{(j)}\} =$$
$$\{(\mathbf{b}, \mathbf{a}) : \underline{n} \in V_{j+1}, children(\underline{n}, x^{(j)}) \neq \underline{n}, \mathbf{b} \in \mathcal{B}(children(\underline{n}, x^{(j)})), \mathbf{a} \in \mathcal{A}(\underline{n})\}.$$

Proof. Any reachable state $\mathbf{z} \in RS$ that has $z^{(j)} = x^{(j)}$ is represented by a path in the MDD that passes through a pair of nodes $\underline{n} \in V_{j+1}$ and $children(\underline{n}, x^{(j)}) \neq \underline{0}$.

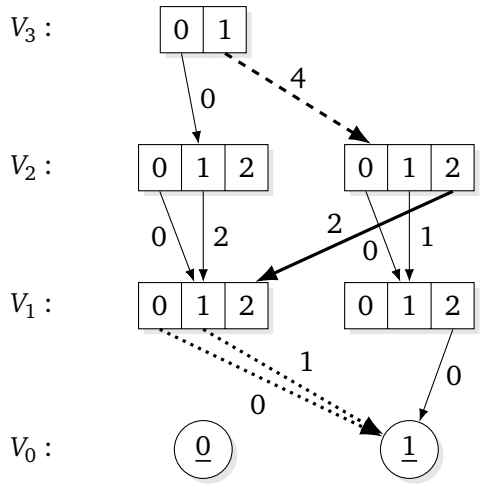


Figure 4.2 The set of all paths having $x^{(1)} = 2^{(1)}$ in the *SharedResource* EDD.

Therefore, some path $\mathbf{a} \in \mathcal{A}(\underline{n})$ must be followed from $\underline{r}$ to reach $\underline{n}$, then some path $\mathbf{b} \in \mathcal{B}(children(\underline{n}, x^{(j)}))$ must be followed from $children(\underline{n}, x^{(j)})$ to $\underline{1}$.

This means all paths from $\underline{r}$ to $\underline{1}$ containing $x^{(j)}$ are of the form $(\mathbf{b}, x^{(j)}, \mathbf{a})$ and the converse also holds. □

Running example 4.3 Figure 4.2 shows all path in the *SharedResource* MDD with $x^{(1)} = 2^{(1)}$.

The single path in the set $\mathcal{A}(\underline{n})$ is dashed, while paths in the set $\mathcal{B}(children(\underline{n}, 2^{(1)}))$ are drawn as dotted edges.

Observation 4.6 If $\underline{n}$ and $\underline{m}$ are distinct nonterminal nodes of a quasi-reduced ordered MDD, $\mathcal{A}(\underline{n}) \cup \mathcal{A}(\underline{m}) = \emptyset$ and $\mathcal{B}(\underline{n}) \neq \mathcal{B}(\underline{m})$.

Proof. We prove the statements indirectly. Let $\mathbf{a} \in \mathcal{A}(\underline{n}) \cup \mathcal{A}(\underline{m})$. If we follow the path $\mathbf{a}$ for $\underline{r}$, we arrive at $\underline{n}$, because $\mathbf{a} \in \mathcal{A}(\underline{n})$. However, we also arrive at $\underline{m}$, because $\mathbf{a} \in Above(\underline{m})$. This is a contradition, since $\underline{n} \neq \underline{m}$, $\mathcal{A}(\underline{n})$ and $\mathcal{A}(\underline{m}$ must be disjoint.

Now suppose that there are $\underline{n}, \underline{m} \in V_N$ such that $\mathcal{B}(\underline{n}) = \mathcal{B}(\underline{m})$. Because the paths $\mathcal{B}(\underline{n})$ describe the subgraph reachable from $\underline{n}$ completely, this means the subgraphs reachable from $\underline{n}$ and $\underline{m}$ are isomorphic. This is impossible, because then the MDD cannot be reduced, thus $\mathcal{B}(\underline{n})$ and $\mathcal{B}(\underline{m})$ must be distinct. □

Observation 4.7 The relation $x^{(j)} \sim^{(j)} y^{(j)}$ can be expressed as

$$x^{(j)} \sim^{(j)} y^{(j)} \iff \{(\underline{n}, children(\underline{n}, x^{(j)})) : \underline{n} \in V_{j+1}\} = \{(\underline{n}, children(\underline{n}, y^{(j)})) : \underline{n} \in V_{j+1}\}.$$

Proof. Let

$$X = \{\hat{\mathbf{z}}^{(j)} : \mathbf{z} \in RS, z^{(j)} = x^{(j)}\}, \qquad Y = \{\hat{\mathbf{z}}^{(j)} : \mathbf{z} \in RS, z^{(j)} = y^{(j)}\}.$$

According to eq. (4.3) on page 31, $x^{(j)} \sim^{(j)} y^{(j)}$ if and only if $X = Y$.

Define

$$X(\underline{n}) = \{\mathbf{b} : (\mathbf{b}, \mathbf{a}) \in X, \mathbf{b} \in \mathcal{A}(\underline{n})\}, \qquad Y(\underline{n}) = \{\mathbf{b} : (\mathbf{b}, \mathbf{a}) \in Y, \mathbf{b} \in \mathcal{A}(\underline{n})\}.$$

$X = Y$ holds precisely when $X(\underline{n}) = Y(\underline{n})$ for all $\underline{n} \in V_{j+1}$. We may notice that $\{X(\underline{n}) \times \mathcal{A}(\underline{n})\}_{\underline{n} \in V_{j+1}}$ and $\{Y(\underline{n}) \times \mathcal{A}(\underline{n})\}_{\underline{n} \in V_{j+1}}$ are partitions of X and Y, respectively, because the $\mathcal{A}$-sets are disjoint for each node.

According to Observation 4.5,

$$X(\underline{n}) = \mathcal{B}(children(\underline{n}, x^{(j)})), \qquad Y(\underline{n}) = \mathcal{B}(children(\underline{n}, y^{(j)})).$$

Thus, $X(\underline{n}) = Y(\underline{n})$ if and only if $children(\underline{n}, x^{(j)}) = children(\underline{n}, y^{(j)})$, because the $\mathcal{B}$-sets are distinct for each node. □

Observation 4.7 can be interpreted as the statement that $x^{(j)} \sim^{(j)} y^{(j)}$ if and only if the MDD edges corresponding to $x^{(j)}$ are always parallel, i.e. from the node $\underline{n}$ they all go to the same node $\underline{m}(\underline{n})$ for all $\underline{n} \in V_{j+1}$.

The macro states can be constructed from the paralell edges in the MDD by partition refinement. This process is performed by Algorithm 4.7.

The key step in partition refinement is in line 15, where the candiate macro state S is split into S_1 and S_2. Edges in S_1 are all parallel and go from $\underline{n}$ to $\underline{m}$, while S_2 is further split. The process is repeated for each node $\underline{n}$ and level V_{j+1} until only parallel macro state candidates remain.

This procedure is based on an idea of Buchholz and Kemper [7], however, we employed partition refinement instead of hashing and and proved correctness of the algorithm formally.

A block Kronecker matrix may be constructed from the decomposed state space by Algorithm 4.4.

$$A = \begin{pmatrix} 1 & 0 & 0 & 2.5 \\ 3 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} A = \{&\{(1,0),(3,1),(4,2),(5,3)\}, \\ &\{(1,1)\}, \\ &\{\}, \\ &\{(2.5,0),(1,2)\}\} \end{aligned}$$

Figure 4.3 Compressed Column Storage of a matrix.

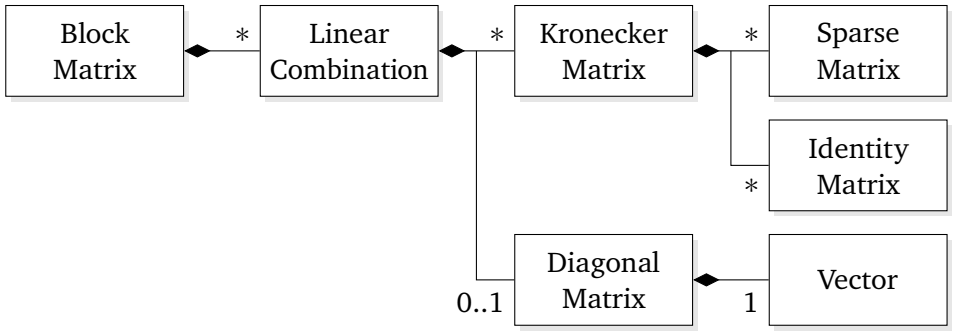


Figure 4.4 Data structure for block Kronecker matrices.

4.3 Matrix storage

Existing linear algebra and matrix libraries, such as [4, 17, 23, 32, 42], usually have unsatisfactory support for operations required in stochastic analysis algorithms with decomposed matrices, for example, multiplications with Kronecker and block Kronecker matrices. Therefore, we have decided to develop a linear algebra framework in C#.NET specifically for stochastic algorithms as a basis of our stochastic analysis framework.

Sparse matrices are stored in Compressed Column Storage (CCS) format, i.e. an array or values and row indices are stored for each column of the matrix, as illustrated in Figure 4.3. This facilitates multiplication from left with row vectors. To reduce pressure on the garbage collector (GC), matrices are vectors are stored in manually allocated and managed memory.

While other sparse matrix formats, such as sliced LAPACK are more amenable to parallel and SIMD processing Kreutzer et al. [27], CCS was selected due to implementation simplicity and the small number of nonzero entries in each column of the matrix, which reduces the potential benefits of SIMD implementations.

Decomposed Kronecker and block Kronecker matrices are stored as algebraic expression trees as shown in Figure 4.4. Matrix multiplication and manipulation algorithms for expression trees are detailed in **??** on page ??.

The expression tree approach allows the use of arbitrary matrix decompositions that can be expressed with block matrices, linear combinations and Kronecker products. The implementation of additional opeartional primitives is also straightforward. The data structure forms a flexible basis for the development of stochastic analysis algorithms with decomposed matrix representations.

Algorithm 4.4 Block Kronecker matrix construction.

Input: State spaces $\widetilde{RS}, \widetilde{RS}^{(j)} RS_x^{(j)}$, transitions T, transition rates Λ, matrices $Q_t^{(j)}, Q_L^{(j)}$
Output: Infinitesimal generator Q

allocate block matrix Q with $\tilde{n} \times \tilde{n}$ blocks
foreach $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \widetilde{RS} \times \widetilde{RS}$ **do**
  Initialize $Q[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}]$ as a linear combination of matrices
  if $\tilde{\mathbf{x}} = \tilde{\mathbf{y}}$ **then**
    for $j \leftarrow 0$ **to** $J-1$ **do**
      if $Q_L^{(j)}[\tilde{x}^{(j)}, \tilde{y}^{(j)}] \neq 0$ **then**
        $I_1 \leftarrow I_{\prod_{f=0}^{j-1} n_{f,x^{(f)}} \times \prod_{h=0}^{j-1} n_{f,x^{(f)}}}, \quad I_2 \leftarrow \mathrm{I}_{\prod_{g=j+1}^{J-1} n_{f,x^{(f)}} \times \prod_{f=j+1}^{J-1} n_{f,x^{(f)}}}$
        $Q[\tilde{\mathbf{x}}, \tilde{\mathbf{x}}] \leftarrow Q[\tilde{\mathbf{x}}, \tilde{\mathbf{x}}] + I_1 \otimes Q_L^{(j)}[\tilde{x}^{(j)}, \tilde{x}^{(j)}] \otimes I_2$
  else if $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is a slmst. at h **then**
    $I_1 \leftarrow I_{\prod_{f=0}^{h-1} n_{f,x^{(f)}} \times \prod_{h=0}^{h-1} n_{f,x^{(f)}}}, \quad I_2 \leftarrow \mathrm{I}_{\prod_{f=f+1}^{J-1} n_{f,x^{(f)}} \times \prod_{f=h+1}^{J-1} n_{f,x^{(f)}}}$
    $Q[\tilde{\mathbf{x}}, \tilde{\mathbf{x}}] \leftarrow Q[\tilde{\mathbf{x}}, \tilde{\mathbf{x}}] + I_1 \otimes Q_L^{(h)}[\tilde{x}^{(h)}, \tilde{x}^{(h)}] \otimes I_2$
  foreach $t \in T_S$ **do**
    Initialize B as an empty Kronecker product
    zeroProduct $\leftarrow$ false
    for $j \leftarrow 0$ **to** $J-1$ **do**
      if $Q^{(j)}[\mathbf{x}, \mathbf{y}] = 0$ **then**
        zeroProduct $\leftarrow$ true
        break
      else if $Q^{(j)}[\mathbf{x}, \mathbf{y}]$ is an identity matrix **then**
        if the last term of B is an indentity matrix $I_{N,N}$ **then**
          Enlarge the last term of B to $I_{N n_{j,x} \times N n_{j,y}}$
        else $B \leftarrow B \otimes Q^{(j)}[\mathbf{x}, \mathbf{y}]$
    if $\neg$*zeroProduct* **then** $Q[\mathbf{x}, \mathbf{y}] \leftarrow Q[\mathbf{x}, \mathbf{y}] + \Lambda(t) B$
allocate block vector $\mathbf{d}$ with $\tilde{n}$ blocks
$\mathbf{d} \leftarrow -Q\mathbf{1}^{\mathrm{T}}$
foreach $\tilde{x} \in \widetilde{RS}$ **do** $Q[\tilde{x}, \tilde{x}] \leftarrow Q[\tilde{x}, \tilde{x}] + \mathrm{diag}\{\mathbf{d}[\tilde{x}]\}$
return Q

Algorithm 4.5 FillIn procedure for matrix construction from EDD state space.

Input: node for target state $\underline{t}$, node for source state $\underline{s}$, target state offset y, source state offset y, transition t, transition rate λ, matrix Q_O

```
1  if level(t) = 0 then
2      if t = 1 ∧ s = 1 then q_O[x, y] ← q_O[x, y] + λ
3  else
4      j ← level(t) − 1
5      foreach y^(j) ∈ RS^(j) do
6          if children(t, y^(j)) = 0 then return
7          Find x^(j) such that x^(j) [t⟩ y^(j)
8          if children(s, x^(j)) = 0 then return
9          FillIn(children(t, y^(j)), children(s, x^(j)),
10                 y + label(t, y^(j)), x + label(t, x^(j)), t, λ, Q_O)
```

Algorithm 4.6 Sparse matric construction from EDD state space.

Input: state space MDD root $\underline{r}$, state space size n, transitions T, transition rate function Λ

Output: generator matrix $Q \in \mathbb{R}^{n \times n}$

```
1  allocate Q_O ∈ ℝ^(n×n)
2  foreach t ∈ T do
3      FillIn(r, r, 0, 0, t, Λ(t), Q_O)
4  d ← −Q_O 1^T
5  return Q_O + diag{d}
```

Algorithm 4.7 Local macro state construction by partition refinement.

Input: Symbolic state space *MDD*

Output: Local macro states $\widetilde{RS}^{(j)}$, $RS_x^{(j)}$

for $j \leftarrow 0$ **to** $J-1$ **do**
  Initialize the empty queue Q
  $Done \leftarrow \{RS^{(j)}\}$
  foreach $\underline{n} \in V_{j+1}$ **do**
    foreach $S \in Done$ **do**
      ENQUEUE(Q, S)
    $Done \leftarrow \emptyset$
    while $\neg$EMPTY(Q) **do**
      $S \leftarrow$ DEQUEUE(Q)
      $S_1 \leftarrow \emptyset$
      $S_2 \leftarrow \emptyset$
      Let x_0 be any element of S
      $\underline{m} \leftarrow children(\underline{n}, x_0)$
      foreach $x \in S \setminus \{x_0\}$ **do**
        if $\underline{m} = children(\underline{n}, x)$ **then** $S_1 \leftarrow S_1 \cup \{x\}$ **else** $S_2 \leftarrow S_2 \cup \{x\}$
      if $S_2 \neq \emptyset$ **then**
        ENQUEUE(Q, S_2)
      $Done \leftarrow Done \cup \{S_1\}$
  $\tilde{n}_j \leftarrow |Done|$
  $\widetilde{RS}^{(j)} \leftarrow \{\tilde{0}^{(j)}, \tilde{1}^{(j)}, \ldots, \widetilde{\tilde{n}_j - 1}^{(j)}\}$
  Each set $S \in Done$ is a local macro state $\widetilde{RS}_x^{(j)}$

References

[1] Falko Bause, Peter Buchholz, and Peter Kemper. “A Toolbox for Functional and Quantitative Analysis of DEDS”. In: *Computer Performance Evaluation: Modelling Techniques and Tools, 10th International Conference, Tools ’98, Palma de Mallorca, Spain, September 14-18, 1998, Proceedings*. Vol. 1469. Lecture Notes in Computer Science. Springer, 1998, pp. 356–359. DOI: 10.1007/3-540-68061-6_32.

[2] Anne Benoit, Brigitte Plateau, and William J. Stewart. “Memory-efficient Kronecker algorithms with applications to the modelling of parallel systems”. In: *Future Generation Comp. Syst.* 22.7 (2006), pp. 838–847. DOI: 10.1016/j.future.2006.02.006.

[3] James T. Blake, Andrew L. Reibman, and Kishor S. Trivedi. “Sensitivity Analysis of Reliability and Performability Measures for Multiprocessor Systems”. In: *SIGMETRICS*. 1988, pp. 177–186. DOI: 10.1145/55595.55616.

[4] BlueBit Software. *.NET Matrix Library 6.1*. Accessed October 26, 2015. URL: http://www.bluebit.gr/NET/.

[5] Peter Buchholz. “Hierarchical Structuring of Superposed GSPNs”. In: *IEEE Trans. Software Eng.* 25.2 (1999), pp. 166–181. DOI: 10.1109/32.761443.

[6] Peter Buchholz, Gianfranco Ciardo, Susanna Donatelli, and Peter Kemper. “Complexity of Memory-Efficient Kronecker Operations with Applications to the Solution of Markov Models”. In: *INFORMS Journal on Computing* 12.3 (2000), pp. 203–222. DOI: 10.1287/ijoc.12.3.203.12634.

[7] Peter Buchholz and Peter Kemper. “Kronecker based matrix representations for large Markov models”. In: *Validation of Stochastic Systems*. Springer, 2004, pp. 256–295.

[8] Peter Buchholz and Peter Kemper. “On generating a hierarchy for GSPN analysis”. In: *SIGMETRICS Performance Evaluation Review* 26.2 (1998), pp. 5–14. DOI: 10.1145/288197.288202.

[9] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. "Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications". In: *IEEE Trans. Computers* 42.11 (1993), pp. 1343–1360. DOI: 10.1109/12.247838.

[10] Piotr Chrzastowski-Wachtel. "Testing Undecidability of the Reachability in Petri Nets with the Help of 10th Hilbert Problem". In: *Application and Theory of Petri Nets 1999, 20th International Conference, ICATPN '99, Williamsburg, Virginia, USA, June 21-25, 1999, Proceedings*. Vol. 1639. Lecture Notes in Computer Science. Springer, 1999, pp. 268–281. DOI: 10.1007/3-540-48745-X_16.

[11] Gianfranco Ciardo, Gerald Lüttgen, and Radu Siminiceanu. *Saturation: an efficient iteration strategy for symbolic state—space generation*. Springer, 2001.

[12] Gianfranco Ciardo, Robert Marmorstein, and Radu Siminiceanu. "The saturation algorithm for symbolic state-space exploration". In: *Int. J. Softw. Tools Technol. Transf.* 8.1 (2006), pp. 4–25. DOI: http://dx.doi.org/10.1007/s10009-005-0188-7.

[13] Gianfranco Ciardo, Jogesh K. Muppala, and Kishor S. Trivedi. "On the Solution of GSPN Reward Models". In: *Perform. Eval.* 12.4 (1991), pp. 237–253. DOI: 10.1016/0166-5316(91)90003-L.

[14] Dániel Darvas. *Szaturáció alapú automatikus modellellenőrző fejlesztése aszinkron rendszerekhez [in Hungarian]*. 1st prize. 2010. URL: http://petridotnet.inf.mit.bme.hu/publications/OTDK2011_Darvas.pdf.

[15] Susanna Donatelli. "Superposed Generalized Stochastic Petri Nets: Definition and Efficient Solution". In: *Application and Theory of Petri Nets 1994, 15th International Conference, Zaragoza, Spain, June 20-24, 1994, Proceedings*. Vol. 815. Lecture Notes in Computer Science. Springer, 1994, pp. 258–277. DOI: 10.1007/3-540-58152-9_15.

[16] Susanna Donatelli. "Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space". In: *Performance Evaluation* 18.1 (1993), pp. 21–36.

[17] Extreme Optimization. *Numerical Libraries for .NET*. Accessed October 26, 2015. URL: http://www.extremeoptimization.com/VectorMatrixFeatures.aspx.

[18] Fault Tolerant Systems Research Group, Budapest University of Technology and Economics. *The Petridotnet webpage*. Accessed October 23, 2015. URL: https://inf.mit.bme.hu/en/research/tools/petridotnet.

[19] Paulo Fernandes, Brigitte Plateau, and William J. Stewart. “Numerical Evaluation of Stochastic Automata Networks”. In: *MASCOTS ’95, Proceedings of the Third International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems, January 10-18, 1995, Durham, North Carolina, USA*. IEEE Computer Society, 1995, pp. 179–183. DOI: 10.1109/MASCOT.1995.378690.

[20] Robert E Funderlic and Carl Dean Meyer. “Sensitivity of the stationary distribution vector for an ergodic Markov chain”. In: *Linear Algebra and its Applications* 76 (1986), pp. 1–17.

[21] Stephen Gilmore and Jane Hillston. “The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling”. In: *Computer Performance Evaluation, Modeling Techniques and Tools, 7th International Conference, Vienna, Austria, May 3-6, 1994, Proceedings*. Vol. 794. Lecture Notes in Computer Science. Springer, 1994, pp. 353–368. DOI: 10.1007/3-540-58021-2_20.

[22] Winfried K. Grassmann. “Transient solutions in markovian queueing systems”. In: *Computers & OR* 4.1 (1977), pp. 47–53. DOI: 10.1016/0305-0548(77)90007-7.

[23] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. Accessed October 26, 2015. 2010. URL: http://eigen.tuxfamily.org.

[24] *International Workshop on Timed Petri Nets, Torino, Italy, July 1-3, 1985*. IEEE Computer Society, 1985. ISBN: 0-8186-0674-6.

[25] Ilse CF Ipsen and Carl D Meyer. “Uniform stability of Markov chains”. In: *SIAM Journal on Matrix Analysis and Applications* 15.4 (1994), pp. 1061–1074.

[26] Peter Kemper. “Numerical Analysis of Superposed GSPNs”. In: *IEEE Trans. Software Eng.* 22.9 (1996), pp. 615–628. DOI: 10.1109/32.541433.

[27] Moritz Kreutzer, Georg Hager, Gerhard Wellein, Holger Fehske, and Alan R. Bishop. “A unified sparse matrix data format for modern processors with wide SIMD units”. In: *CoRR* abs/1307.6209 (2013). URL: http://arxiv.org/abs/1307.6209.

[28] Francesco Longo and Marco Scarpa. “Two-layer Symbolic Representation for Stochastic Models with Phase-type Distributed Events”. In: *Intern. J. Syst. Sci.* 46.9 (2015), pp. 1540–1571. DOI: 10.1080/00207721.2013.822940.

[29] Marco Ajmone Marsan. “Stochastic Petri nets: an elementary introduction”. In: *Advances in Petri Nets 1989, covers the 9th European Workshop on Applications and Theory in Petri Nets, held in Venice, Italy in June 1988, selected papers*. Vol. 424. Lecture Notes in Computer Science. Springer, 1988, pp. 1–29. DOI: 10.1007/3-540-52494-0_23.

[30] Marco Ajmone Marsan, Gianfranco Balbo, Andrea Bobbio, Giovanni Chiola, Gianni Conte, and Aldo Cumani. “The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets”. In: *IEEE Trans. Software Eng.* 15.7 (1989), pp. 832–846. DOI: 10.1109/32.29483.

[31] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. “A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems”. In: *ACM Trans. Comput. Syst.* 2.2 (1984), pp. 93–122. DOI: 10.1145/190.191.

[32] Math.NET. *Math.NET Numerics webpage*. Accessed October 26, 2015. URL: http://numerics.mathdotnet.com/.

[33] Tadao Murata. “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580.

[34] M. Neuts. “Probability distributions of phase type”. In: *Liber Amicorum Prof. Emeritus H. Florin*. University of Louvain, 1975, pp. 173–206.

[35] RJ Plemmons and A Berman. *Nonnegative matrices in the mathematical sciences*. Academic Press, New York, 1979.

[36] S. Rácz, Á. Tari, and M. Telek. “MRMSolve: Distribution estimation of Large Markov reward models”. In: *Tools 2002*. Springer, LNCS 2324, 2002, pp. 72–81.

[37] S. Rácz and M. Telek. “Performability Analysis of Markov Reward Models with Rate and Impulse Reward”. In: *Int. Conf. on Numerical solution of Markov chains*. 1999, pp. 169–187.

[38] A. V. Ramesh and Kishor S. Trivedi. “On the Sensitivity of Transient Solutions of Markov Models”. In: *SIGMETRICS*. 1993, pp. 122–134. DOI: 10.1145/166955.166998.

[39] Andrew Reibman, Roger Smith, and Kishor Trivedi. “Markov and Markov reward model transient analysis: An overview of numerical approaches”. In: *European Journal of Operational Research* 40.2 (1989), pp. 257–267.

[40] Pierre Roux and Radu Siminiceanu. “Model Checking with Edge-valued Decision Diagrams”. In: *Second NASA Formal Methods Symposium - NFM 2010, Washington D.C., USA, April 13-15, 2010. Proceedings*. Vol. NASA/CP-2010-216215. NASA Conference Proceedings. 2010, pp. 222–226.

[41] *Advances in Petri Nets 1987, covers the 7th European Workshop on Applications and Theory of Petri Nets, Oxford, UK, June 1986*. Vol. 266. Lecture Notes in Computer Science. Springer, 1987. ISBN: 3-540-18086-9.

[42] Conrad Sanderson. “Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments”. In: (2010).

[43] Williams J Stewart. *Introduction to the numerical solutions of Markov chains*. Princeton Univ. Press, 1994.

[44] Enrique Teruel, Giuliana Franceschinis, and Massimiliano De Pierro. "Well-Defined Generalized Stochastic Petri Nets: A Net-Level Method to Specify Priorities". In: *IEEE Trans. Software Eng.* 29.11 (2003), pp. 962–973. DOI: 10.1109/TSE.2003.1245298.