Using Structured Modelling for Efficient Performance Prediction of Parallel Systems

Markus Siegle

Computer Science Department (IMMD VII), Universität Erlangen-Nürnberg,
Martensstraße 3, 91058 Erlangen, Germany, siegle@informatik.uni-erlangen.de

Abstract: A method for analyzing parallel systems with the help of structured Markovian models is presented. The overall model is built from interdependent submodels, and symmetries are exploited automatically by grouping similar submodels in classes. During model analysis, this leads to a state space reduction, based on the concept of exact lumpability.

1. Introduction

The high complexity of today's parallel computers — due to the large number of concurrently active and mutually dependent hardware and software components — makes them difficult to understand for human users. Developing efficient software for this class of machines is therefore very time-consuming and expensive. The resulting performance often fails to meet expectations, therefore requiring major re-implementation. Event-based performance models can be used to alleviate this problem. They help humans to understand the dynamic behaviour of parallel systems by abstracting the overwhelming number of details of the real world to essentials. They are especially useful for *predicting the performance of not yet implemented systems*, which helps to avoid expensive implementation work. Models assist application programmers in writing more efficient parallel code. With the advent of the modern parallel computer generation, modelling theory has an increased practical relevance. The development of efficient techniques for the modelling of parallel systems is therefore an important contribution to the understanding and acceptance of such systems.

Markovian models are widely used for the modelling of parallelism because they are amenable to numerical analysis. For their specification such different modelling paradigms as queueing networks, stochastic Petri nets and stochastic automata can be used. Tools exist [1, 2] which support the generation of high-level models which are then translated into an underlying continuous time Markov chain. The solution of the Markov chain yields probability values from which performance and reliability measures of the modelled system can be derived. *Steady-state analysis* is employed for studying the long-range behaviour of systems reaching an equilibrium, whereas the system's development over a limited period of time is investigated by means of *transient analysis*. In this paper, only steady-state analysis will be considered.

The major problem in Markovian modelling of complex real systems is the high number of model states. Memory space needed for the model's stochastic generator matrix Q often exceeds the available capacity, and computation time for solving the Markov chain (i.e. solving the system of linear equations $pQ = 0$, where p is the vector of steady-state probabilities) can make model analysis prohibitively expensive.

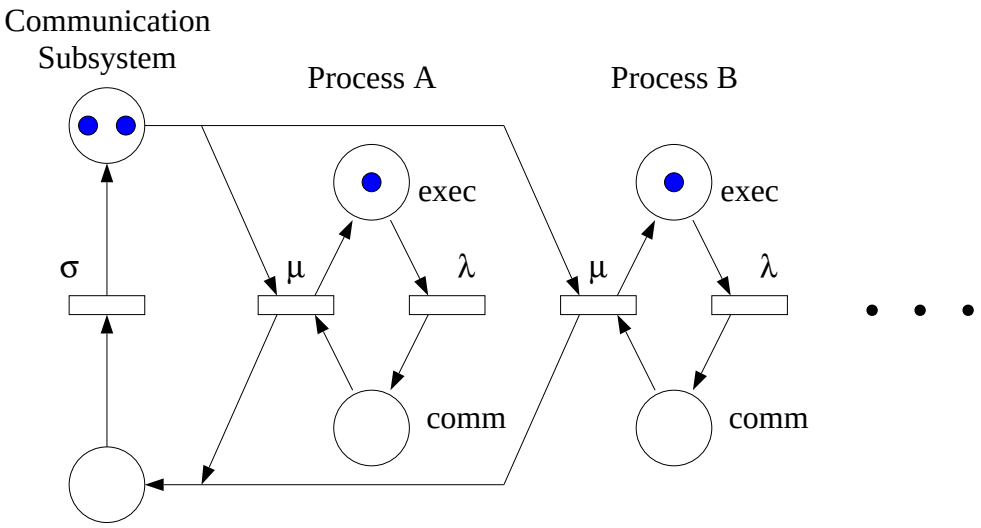

Figure 1: Simple Parallel Processing Model (Stochastic Petri Net)

As a simple example, consider the stochastic Petri net model shown in Fig. 1. It represents two processes (each is assumed to run independently on a separate node of a multiprocessor) which are either executing work (*exec*) or attempting to submit a message to a communication subsystem (*comm*). There are two message buffer places, and message submission is only possible if there is such a buffer available. A process willing to submit a message is blocked until a buffer is available. The durations of the *exec* and *comm* activities are exponentially distributed with rates λ and μ respectively. The communication subsystem processes messages at rate σ. This Petri net has $3 \times 2^2 = 12$ reachable markings, i.e. the underlying continuous time Markov chain (CTMC) has 12 states. For a general case with n instead of 2 processes involved (as indicated by the dots in Fig. 1) but still with only two buffers, the number of states is given by the expression 3×2^n. We see that, although being very simple, the example has a state space which grows exponentially with respect to the number of components involved. This phenomenon is quite typical for models of parallel systems.

Among the techniques that have been considered in order to overcome the state space explosion problem are model decomposition [3], model simplification [4] and structured modelling. In this paper, we focus on the latter approach.

2. Structured Modelling

Structured modelling is based on information about the modular structure of the (existing or projected) real system to be modelled. This makes it superior to model decomposition, where an overall model must be decomposed into submodels whose identification often constitutes a serious problem. In structured modelling the question of how to decompose a large overall model never arises. It is implicitly answered by the structure of the real system in the following way: For each of the modules of the real system there is a corresponding submodel which may be specified in isolation. During model analysis, the structure of the model is exploited, making it possible to analyze complex models whose solution would otherwise not be feasible. Carrying out modelling in a structured way brings recognized advantages from

other areas such as software engineering. Modularity makes it easier for humans to understand models, and submodels can be replaced very easily without affecting the specification of other components, resulting in a great flexibility. Among published approaches to structured modelling, the following are of particular relevance:

- In the work of Plateau et al., stochastic automata networks are used for model specification [5, 2]. Information on the interdependence between automata is given by synchronizing events and functional rates (rates dependent on the global state). It is shown that the generator matrix of the overall model can be described by a tensor expression involving only small submodel matrices. The iterative numerical solution (power method) of the model can be carried out in a *distributed* fashion on this tensor descriptor. This technique saves a lot of memory space, thus making large models tractable. Parallelization of the scheme is possible.
- In the work of Buchholz [6], the tensor descriptor also plays an important role. He uses two-layer hierarchical multi-paradigm models. A high-level model defines the flow of entities (customers, tokens) between the submodels, resulting in an entity-oriented view (as opposed to the state-oriented approach of Plateau).

Our own intention is to develop a framework which combines features of the above mentioned approaches. While we allow multiple paradigms to be used for submodel specification, we preserve the state-oriented view of Plateau. This avoids the restriction of submodel interdependence to the flow of entities, allowing arbitrary interdependence instead. A new feature is the grouping of submodels into classes. Submodel classes are introduced in view of automatic symmetry exploitation, and it will be shown in the course of this paper (see Section 3) that this can lead to significant state space reduction.

We return to the parallel processing example of Section 1, which will now be described by a structured model. In Fig. 2, three submodels (SM1, SM2 and SM3) are shown which represent the system components. The communication subsystem (SM1) is specified by an M/M/1/2 queue, i.e. it consists of an exponential server with rate σ and a finite buffer with a total capacity of 2 customers (including the one in service). The two processes are represented by the (identical) stochastic automata submodels SM2 and SM3. This stochastic automaton has two states and two possible state transitions: The transition from *exec* to *comm* is a purely local event which takes place at rate λ. The reverse transition from *comm* to *exec*, however, can only take place synchronously with an arrival to the communication subsystem SM1. This is an example of a synchronizing event. It is labelled *e* in Fig. 2.

Since both (in general all n) process submodels are identical, we say that they belong to the same *submodel class* (indicated by the dashed boxes in Fig. 2). In general, all similar submodels constitute a class. The semantics of the synchronizing event *e* is to be chosen here so that only one submodel per class is participating in the event. For the two process case, this means that the event *e* synchronizes either the pair of submodels (SM1, SM2) or the pair (SM1, SM3). There is another possible semantics for synchronizing events so that all submodels within a class must participate (this would synchronize all three submodels SM1, SM2 and SM3 which is not the intended behaviour).

We will now show how the generator matrix of the overall model can be described by smaller submodel matrices and the use of tensor operations. The matrices associated with SM1 are

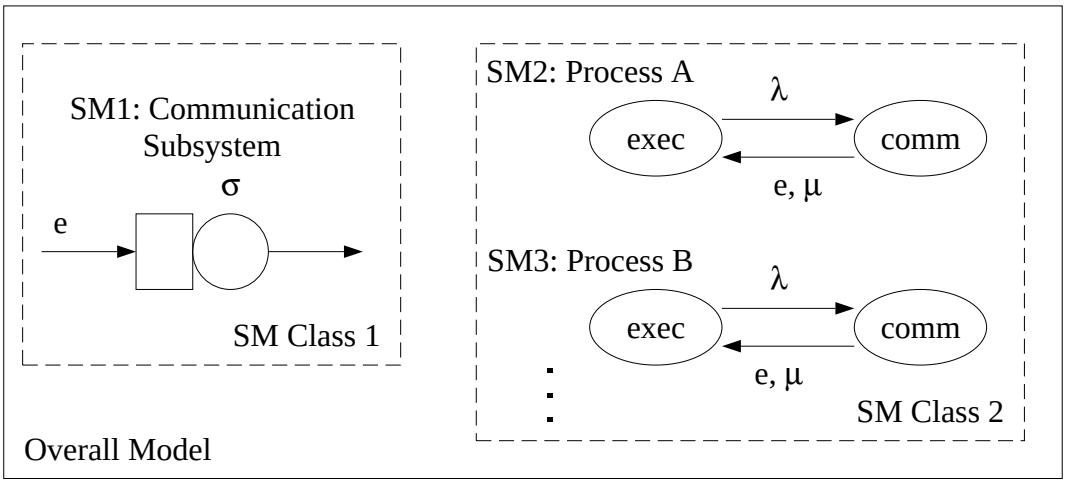

Figure 2: Structured Model of Simple Parallel Processing

all of dimension 3×3 (since SM1 has three possible states) and are given by

$$Q_l^{(1)} = \begin{bmatrix} 0 & 0 & 0 \\ \sigma & -\sigma & 0 \\ 0 & \sigma & -\sigma \end{bmatrix}, \; Q_e^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \; Q_{e,n}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1}$$

Here, $Q_l^{(1)}$ contains the rates of transitions local to submodel SM1, namely the service of messages which takes place at rate σ, provided that at least one message is in the buffer of the communication subsystem. Matrix $Q_e^{(1)}$ describes how SM1 is affected by the synchronizing event e, the arrival of a message. Matrices denoted by $Q_{e,n}^{(i)}$ will be needed to correct the entries on the diagonal of the overall generator matrix. In general, their diagonal elements are equal to the row sums of the corresponding $Q_e^{(i)}$, and all off-diagonal entries equal zero. In this particular case, all non-zero entries of $Q_e^{(1)}$ are equal to unity, because the (active) rate of transition e will be associated with SM2 and SM3, while SM1 is participating in the synchronization passively. The matrices for submodels SM2 and SM3 are given by

$$Q_l^{(i)} = \begin{bmatrix} -\lambda & \lambda \\ 0 & 0 \end{bmatrix}, \; Q_e^{(i)} = \begin{bmatrix} 0 & 0 \\ \mu & 0 \end{bmatrix}, \; Q_{e,n}^{(i)} = \begin{bmatrix} 0 & 0 \\ 0 & \mu \end{bmatrix}, \; i = 2, 3 \tag{2}$$

The generator matrix Q of the overall model can now be built from these submatrices by using the operators $\oplus$ and $\otimes$ (tensor sum and tensor product, see appendix).

$$\begin{aligned} Q = & \, Q_l^{(1)} \oplus Q_l^{(2)} \oplus Q_l^{(3)} \\ & + Q_e^{(1)} \otimes Q_e^{(2)} \otimes I_2 - Q_{e,n}^{(1)} \otimes Q_{e,n}^{(2)} \otimes I_2 \\ & + Q_e^{(1)} \otimes I_2 \otimes Q_e^{(3)} - Q_{e,n}^{(1)} \otimes I_2 \otimes Q_{e,n}^{(3)} \end{aligned} \tag{3}$$

The full structure of matrix Q is depicted in Fig. 3.

Diagonal entries denoted by Σ are equal to the negative row sum. The tensor descriptor form of Q in the general case with n process submodels is a straightforward extension of the expression in Eq. (3). It is given by

$$Q = \bigoplus_{i=1}^{n+1} Q_l^{(i)} + \sum_{i=2}^{n+1} Q_e^{(1)} \otimes I_2 \otimes \cdots \otimes Q_e^{(i)} \otimes \cdots \otimes I_2 - \sum_{i=2}^{n+1} Q_{e,n}^{(1)} \otimes I_2 \otimes \cdots \otimes Q_{e,n}^{(i)} \otimes \cdots \otimes I_2 \tag{5}$$

$$Q = \begin{bmatrix}
\Sigma & \lambda & \lambda & & & & & & & & & \\
 & \Sigma & & \lambda & \mu & & & & & & & \\
 & & \Sigma & \lambda & \mu & & & & & & & \\
 & & & \Sigma & & \mu & \mu & & & & & \\
\sigma & & & & \Sigma & \lambda & \lambda & & & & & \\
 & \sigma & & & & \Sigma & & \lambda & \mu & & & \\
 & & \sigma & & & & \Sigma & \lambda & \mu & & & \\
 & & & \sigma & & & & \Sigma & & \mu & \mu & \\
 & & & & \sigma & & & & \Sigma & \lambda & \lambda & \\
 & & & & & \sigma & & & & \Sigma & & \lambda \\
 & & & & & & \sigma & & & & \Sigma & \lambda \\
 & & & & & & & \sigma & & & & \Sigma
\end{bmatrix}$$

Figure 3: Generator Matrix Q for the $n = 2$ Case

In this section we have demonstrated some important features of structured modelling. Submodels which correspond to the various components of the real system to be modelled can be specified individually. Different modelling paradigms such as stochastic automata, queueing models and stochastic Petri nets can be used within one overall model. Since the explicit construction of the model's state space may be avoided, the memory space for the (potentially very large) stochastic generator matrix is not needed.

3. Exploiting Symmetries

While structured description and analysis help to make large models tractable, large state spaces still make analysis expensive, so the aim of reducing the state space remains. In this section we describe how the cardinality of the state space may be reduced drastically by taking advantage of *symmetries* in the model. Since many modern parallel systems consist of a high number of similar components (processors, memory modules, busses), and since parallel application programs often involve replicated processes, models of such systems will exhibit symmetries, and therefore this is an issue of great practical relevance.

The idea of state space reduction through symmetry exploitation is by no means new. It is a technique which has been used very often, but usually in a rather specialized way, tailored to the particular case study at hand. Two recent approaches to a general modelling procedure in which symmetry exploitation is an essential part are the following:

- In the context of stochastic Petri nets it has been shown how symmetry can be exploited by folding identical subnets of a GSPN, thereby obtaining a concise coloured GSPN representation. The definition of an equivalence relation on the set of coloured net markings (the model's state space) and the concept of symbolic reachability graph (SRG) [7] result in a reduction of the model's state space. In [8] an example for the SRG technique is given where state space reduction from exponential to linear growth (with respect to the degree of parallelism) is achieved.
- For stochastic activity networks — a particular class of stochastic Petri nets — an algorithm for reducing the state space in the presence of replicated subsystems has been developed [9].

Using structured modelling, symmetries are known from model structure. In our framework of structured models, with submodel classes containing a number of similar submodels, there are states which are symmetric. Such states have the same steady-state probability and it is redundant to treat them individually. The idea of symmetry exploitation is to combine all submodels of a class and replace them by a single reduced model in order to avoid this redundancy. During the construction of the reduced model, only one state is chosen as a representative for each set of symmetric states. We will now show how our approach to structured modelling, and in particular the notion of submodel classes, provides a general framework for the exploitation of symmetries.

Submodel Classes and Markov Chain Lumpability

Suppose we have a structured model consisting of n submodels $SM_1, \ldots, SM_n$. Furthermore, let this set of submodels be the union of c classes such that all similar submodels belong to the same class, i.e. a class contains multiple copies of the same submodel. Let class i contain n_i submodels, each having s_i states. We have the relation $\Sigma_{i=1}^{c} n_i = n$, and the number of states s of the overall model is given by $s = \Pi_{i=1}^{c} s_i^{n_i}$ (not all states must be actually reachable).

Let us now focus our attention to class i, and without loss of generality assume that the submodels in this class are denoted $SM_1, \ldots, SM_{n_i}$. The combined state space S_i of all submodels of class i has $s_i^{n_i}$ states, where every state is an n_i-tuple with elements from $\{0, \ldots, s_i - 1\}$. We consider two states $x = (x_1, \ldots, x_{n_i})$ and $y = (y_1, \ldots, y_{n_i})$ symmetric if y is a permutation of x, i.e. if there exists a permutation matrix P of dimension n_i such that $y = xP$. The state space S_i is now partitioned in such a way that all symmetric states are in the same partition. This partition is clearly an equivalence relation, and its construction guarantees that the sum of the transition rates from a given state to all states in another partition is the same for all states within the same partition. Therefore the Markov process with state space S_i is lumpable [10] with respect to this partition. A reduced model can then be built which contains only one state per set of symmetric states.

In general, the generator matrix $\hat{Q}$ of the lumped Markov chain is obtained formally by right-multiplying the original generator matrix Q with a projection matrix V, and by left-multiplying with a selection matrix U.

$$\hat{Q} = UQV \tag{6}$$

In our framework we have a special case of lumpability. Within each class, states are in lexicographical ordering as a result of the tensor operations. Therefore, when lumping the combined state space of submodel class i, the structure of the matrices U and V depends only on n_i and s_i. Thus, these matrices are known a priori, and we will denote them $U_{s_i}^{n_i}$ and $V_{s_i}^{n_i}$. The dimensions of matrices $U_{s_i}^{n_i}$ and $V_{s_i}^{n_i}$ are $r \times s_i^{n_i}$ and $s_i^{n_i} \times r$, where $r = \binom{n_i + s_i - 1}{s_i - 1}$. Lumping symmetric states in the combined stochastic process of class i reduces the state space of this process from $s_i^{n_i}$ to $\binom{n_i + s_i - 1}{s_i - 1}$.

Returning to the example shown in Fig. 2, we observe that in the combined state space of submodels SM2 and SM3, S_2, states $(exec, comm)$ and $(comm, exec)$ are symmetric, i.e. S_2 will be partitioned as $S_2 = \{(exec, exec)\} \cup \{(exec, comm), (comm, exec)\} \cup \{(comm, comm)\}$. The combination of SM2 and SM3 is replaced by a reduced stochastic

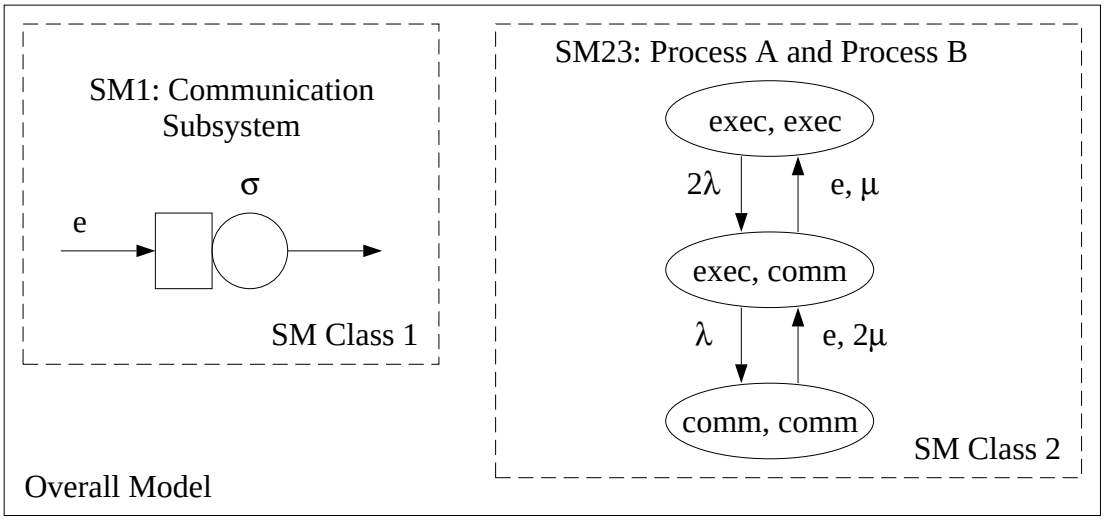

Figure 4: Reduced Structured Model of Simple Parallel Processing

automaton with only 3 states (as opposed to 4) as shown in Fig. 4. For the overall model, this means that the number of states is reduced from $3 \times 2^2 = 12$ to $3 \times \binom{3}{1} = 3 \times 3 = 9$. The matrices associated with the reduced model can be computed automatically. Local transitions of the combination of SM2 and SM3 are described by $\hat{Q}_l^{(23)}$, given by

$$\begin{aligned}\hat{Q}_l^{(23)} &= U_2^2 \left(Q_l^{(2)} \oplus Q_l^{(3)} \right) V_2^2 \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2\lambda & \lambda & \lambda & 0 \\ 0 & -\lambda & 0 & \lambda \\ 0 & 0 & -\lambda & \lambda \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -2\lambda & 2\lambda & 0 \\ 0 & -\lambda & \lambda \\ 0 & 0 & 0 \end{bmatrix}\end{aligned} \tag{7}$$

Likewise, for the matrices describing the synchronization with SM1 by the event e, we obtain $\hat{Q}_e^{(23)} = U_2^2 \left(Q_e^{(2)} \oplus Q_e^{(3)} \right) V_2^2$ and $\hat{Q}_{e,n}^{(23)} = U_2^2 \left(Q_{e,n}^{(2)} \oplus Q_{e,n}^{(3)} \right) V_2^2$.

4. Conclusion

In this paper, we have discussed significant advantages which arise if modelling is carried out in a structured fashion. We have presented a framework which is based on the tensor descriptor approach of Plateau, but is extended in such a way that symmetries are recognized and exploited in an automizable way. Our framework, a complete formal definition of which is yet to be given, is kept general enough to allow multiple paradigms to be employed for the specification of submodels. Exact lumpability is the theoretical background for symmetry exploitation, which is carried out at the level of submodel classes. The degree to which the state space is reduced is predictable — it depends on the number of submodels within a class and on the cardinality of the state space of these submodels — and a reduction of the state space of a submodel class implies the reduction of the overall state space by the same factor. So far there is no existing tool support for our method, and this is clearly a challenge for future work.

References

[1] G. Ciardo and J. Muppala. *Manual for the SPNP Package Version 3.1*. Duke University, October 1991.

[2] B. Plateau, J.-M. Fourneau, and K.-H. Lee. PEPS: A Package for Solving Complex Markov Models of Parallel Systems. In *Proceedings of the 4th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 341–360, Palma (Mallorca), September 1988.

[3] G. Ciardo and K. Trivedi. Solution of Large GSPN Models. In W. Stewart, editor, *Numerical Solution of Markov Chains*, pages 565–595. Marcel Dekker, New York, Basel, Hong Kong, 1991.

[4] C. Simone and M. A. Marsan. The Application of EB-Equivalence Rules to the Structural Reduction of GSPN Models. *Journal of Parallel and Distributed Computing*, 15(3):296–302, July 1992.

[5] B. Plateau. On the Synchronization Structure of Parallelism and Synchronization Models for Distributed Algorithms. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 147–154, Austin, TX, August 1985. ACM.

[6] P. Buchholz. Numerical Solution Methods Based on Structured Descriptions of Markovian Models. In G. Balbo and G. Serazzi, editors, *5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 242–258, Torino, February 1991.

[7] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On Well-Formed Coloured Nets and their Symbolic Reachability Graph. In *Proceedings of the 11th International Conference on Application and Theory of Petri Nets*, pages 387–410, Paris, June 1990. reprinted in High-level Petri Nets.

[8] M. Siegle. On Efficient Markovian Modelling. In *Proc. QMIPS Workshop on Stochastic Petri Nets*, pages 213–225, Sophia Antipolis, November 1992.

[9] W. Sanders and J. Meyer. Reduced Base Model Construction Methods for Stochastic Activity Networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, January 1991.

[10] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

Appendix — Basic Tensor Operations

Let $M(r,c)$ be the set of matrices with r rows and c columns. The tensor product (Kronecker product) of two matrices $A \in M(r_A, c_A)$ and $B \in M(r_B, c_B)$ is the matrix $C \in M(r_A r_B, c_A c_B)$ such that

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1c_A}B \\ a_{21}B & & & \\ \vdots & & & \vdots \\ a_{r_A 1}B & & \dots & a_{r_A c_A}B \end{bmatrix}$$

For example,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & a_{13}b_{11} & a_{13}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & a_{13}b_{21} & a_{13}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} & a_{23}b_{11} & a_{23}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} & a_{23}b_{21} & a_{23}b_{22} \end{bmatrix}$$

The tensor sum of two square matrices $A \in M(d_A, d_A)$ and $B \in M(d_B, d_B)$ is defined as $A \oplus B = A \otimes I_{d_B} + I_{d_A} \otimes B$, where I_d denotes an identity matrix of dimension d.

For example,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & b_{12} & a_{12} & & a_{13} & \\ b_{21} & a_{11}+b_{22} & & a_{12} & & a_{13} \\ a_{21} & & a_{22}+b_{11} & b_{12} & a_{23} & \\ & a_{21} & b_{21} & a_{22}+b_{22} & & a_{23} \\ a_{31} & & a_{32} & & a_{33}+b_{11} & b_{12} \\ & a_{31} & & a_{32} & b_{21} & a_{33}+b_{22} \end{bmatrix}$$