Markovian Performance Modelling of Parallel Systems

Markus Siegle
Computer Science Department (IMMD VII), University of Erlangen-Nürnberg
Martensstraße 3, D-8520 Erlangen, Germany, email: siegle@informatik.uni-erlangen.de

Abstract: Techniques for the efficient modelling of modern parallel systems are discussed. Structured modelling and the use of tensor algebra make it possible to analyze models which are otherwise not tractable. A key point is the exploitation of model symmetries and its relation to the SRG technique developed in the field of coloured stochastic Petri nets.

1. Introduction

Parallel systems have a high complexity and are therefore often difficult for human users to understand. The complexity is due to the high number of concurrently active and mutually dependent hardware and software components in today's parallel computers. Event-based models help humans to understand the dynamic behaviour of parallel systems by abstracting the overwhelming number of details of the real world to essentials. Timed models 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 modern parallel systems, 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 (restricted to memoryless service time distributions) are widely used for the modelling of parallelism because they are amenable to numerical analysis. For their specification, such different modelling paradigms as stochastic automata, stochastic Petri nets, queueing networks and stochastic graph models can be used. Tools exist [1, 2, 3] to support the generation of high-level models which are then transformed into an underlying continuous time Markov chain (CTMC). The steady-state solution of the Markov chain yields probability values from which performance measures of the modelled system can be derived. Reliability measures are obtained from the transient solution of the Markov chain.

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 often exceeds the capacity of today's computers, and computation time for model analysis can make the solution prohibitively expensive. Techniques to overcome this state space explosion problem include model decomposition, model transformation for the sake of simplification, and structured modelling. This paper is focussed on the latter.

2. Structured Modelling

In contrast to model decomposition, where an overall model is decomposed into submodels whose identification often constitutes a serious problem, structured modelling is based on knowledge given a priori by the modular structure of the real system. Therefore the question how to

decompose the overall model never arises. It is implicitly answered by the structure of the real system. During model analysis, the structure of the model is exploited, making it possible to analyze complex models whose solution would otherwise not be feasible.

Structured modelling has two main advantages: First, the explicit construction of the model's state space may be avoided, thereby saving the memory space for the (potentially very large) stochastic generator matrix. The solution of the model can be carried out in a *distributed* fashion by applying iterative methods involving only small submodel generator matrices [4]. Second, the cardinality of the state space may be reduced drastically by taking advantage of *symmetries* in the model (see Section 3). This is an important feature when modelling today's parallel systems with a high number of identical components.

The concept of structured modelling will now be demonstrated by a simple example. In Figure 1 two stochastic Petri net submodels and their corresponding reachability graphs are shown. For the scope of this example, there is no particular meaning associated with the Petri net submodels — the reader may think of them as two interacting components of some real system. Submodel X_1 has two reachable markings (states), whereas submodel X_2 has three states (beware of the arc multiplicities). Transition firing times are exponentially distributed with rates r_{ij} and s_{ij}. Assuming infinite-server behaviour for transition T_{22}, the relation $s_{12} = 2s_{23}$ holds. Submodel X_1 is independent of submodel X_2, but not vice versa. Transition T_{21} of X_2 is only enabled if there is a token in the lower place of X_1, i.e. its firing is dependent on the condition $\langle X_1 = 2 \rangle$.

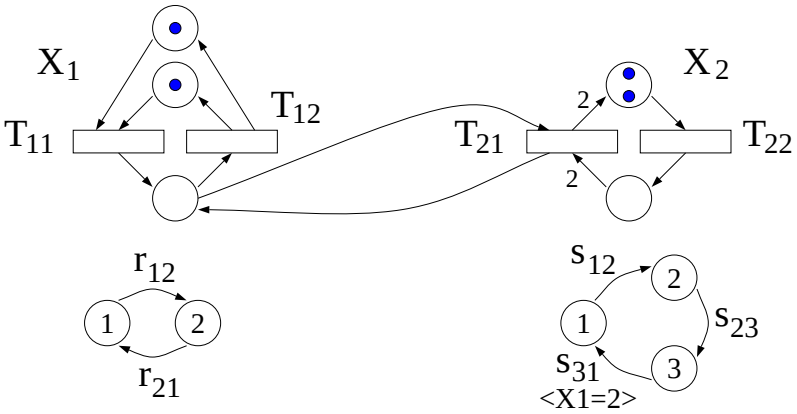

Figure 1: Two dependent Petri net submodels

The overall model, resulting from the combination of submodels X_1 and X_2, has a state space equal to the Cartesian product of the submodel state spaces. For the computation of its steady-state probabilities, a linear homogeneous system of equations $pT = 0$ has to be solved, where T denotes the stochastic generator matrix of the overall model. T can be expressed in terms of smaller matrices: generator, indicator and transition matrices associated with submodels X_1 and X_2. The stochastic generator matrix R of X_1 and the matrix S, which contains those transitions in X_2 which are independent of the state of X_1, are given by (asterisks denote 0 entries)

$$R = \begin{bmatrix} -r_{12} & r_{12} \\ r_{21} & -r_{21} \end{bmatrix} \qquad S = \begin{bmatrix} -s_{12} & s_{12} & * \\ * & -s_{23} & s_{23} \\ * & * & * \end{bmatrix}$$

The dependence of the transition from state 3 to state 1 in submodel X_2 is expressed by the two following matrices R_a and S_a. R_a indicates that the condition for the dependent transition

in X_2 is fulfilled in state 2, and S_a contains the rate of the dependent transition and a negative entry in the diagonal to compensate the row sum accordingly.

$$R_a = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad S_a = \begin{bmatrix} * & * & * \\ * & * & * \\ s_{31} & * & -s_{31} \end{bmatrix}$$

The overall stochastic process has $2 * 3 = 6$ states. Its stochastic generator matrix, with the help of matrices R, S, R_a and S_a associated with the submodels X_1 and X_2, and the use of the tensor operations $\oplus$ (tensor sum) and $\otimes$ (tensor product) [5], can be expressed as

$$T = (R \oplus S) + (R_a \otimes S_a)$$

$$= \begin{bmatrix} \Sigma & s_{12} & & r_{12} & & \\ & \Sigma & s_{23} & & r_{12} & \\ & & \Sigma & & & r_{12} \\ r_{21} & & & \Sigma & s_{12} & \\ & r_{21} & & & \Sigma & s_{23} \\ & & r_{21} & & & \Sigma \end{bmatrix} + \begin{bmatrix} * & & & & & \\ & * & & & & \\ & & * & & & \\ & & & * & & \\ & & & & * & \\ & & & s_{31} & & -s_{31} \end{bmatrix}$$

The entries in the diagonal denoted by Σ represent the negative row sum, i.e. $\Sigma = \sum_{j \neq i} t_{ij}$ in row i.

In our small example there is a dependence of one submodel's behaviour on the state of another submodel. However, different types of dependences are possible, such as the synchronization of state transitions in different submodels, or the flow of a token from one Petri net submodel to another, thereby changing simultaneously the states of both submodels involved. In [4, 6, 7] the concept of structured modelling with the help of tensor algebra is discussed in the special contexts of stochastic automata, hierarchical multi-paradigm models (queueing network and stochastic Petri net submodels) and superposed stochastic automata. In a general context, the generator matrix of an overall model consisting of n interdependent submodels can be written as

$$T = \bigoplus_{i=1}^{n} R_i + \sum_a \bigotimes_{i=1}^{n} R_{ia} - \sum_a \bigotimes_{i=1}^{n} R'_{ia}$$

where the summations are over all dependences a. For the solution of the linear homogeneous system of equations $pT = 0$, direct methods are usually not appropriate because of the sparsity of the matrix T and the fill-in arising during modification of T. Therefore iterative methods are applied. It is known that the structure of T allows a very efficient implementation of iterative methods. We will demonstrate this with a brief discussion of the power method. For this method, the continuous time Markov process is discretized, resulting in the iteration scheme

$$p_{k+1} = p_k(Id + \alpha T) \quad ; \quad \frac{1}{\alpha} > \max_i \{|t_{ii}|\}$$

where Id is an identity matrix of appropriate size. In each step of the iteration the following expression has to be evaluated

$$\alpha p_k T = \alpha p_k \left(\bigoplus_{i=1}^{n} R_i + \sum_a \bigotimes_{i=1}^{n} R_{ia} - \sum_a \bigotimes_{i=1}^{n} R'_{ia} \right)$$

The product of the vector p_k and the tensor sum can be computed efficiently by writing the tensor sum as an ordinary matrix sum of tensor products and expressing each element of the sum as a permutation of a so-called normal factor.

$$p_k \bigoplus_{i=1}^{n} R_i = p_k \sum_{i=1}^{n} Id_{l_i} \otimes R_i \otimes Id_{u_i} = p_k \sum_{i=1}^{n} P_{\sigma_i}^T (Id_{l_i u_i} \otimes R_i) P_{\sigma_i}$$

$$= \sum_{i=1}^{n} p_k^{\sigma_i^T} \begin{bmatrix} R_i & & & & \\ & R_i & & & \\ & & . & & \\ & & & . & \\ & & & & R_i \end{bmatrix} P_{\sigma_i}$$

Normal factors consist of one block which is replicated along the diagonal of the matrix. The multiplication of a vector with a normal factor only requires the combination of subvectors with the diagonal blocks of the matrix. Furthermore, permutations can be realized by address transformation. The multiplication of the vector p_k and the tensor products can also be done efficiently by writing the tensor product as an ordinary matrix product of permuted normal factors.

$$p_k \bigotimes_{i=1}^{n} R_{ia} = p_k \prod_{i=1}^{n} Id_{l_i} \otimes R_{ia} \otimes Id_{u_i} = \ \ldots =$$

$$= p_k P_{\sigma_1}^T (Id_{l_1 u_1} \otimes R_{1a}) P_{\sigma_1} P_{\sigma_2}^T (Id_{l_2 u_2} \otimes R_{2a}) P_{\sigma_2} \ldots P_{\sigma_n}^T (Id_{l_n u_n} \otimes R_{na}) P_{\sigma_n}$$

It is an important advantage of this method that the generator matrix T of the overall model never needs to be generated explicitly. Since this matrix can be extremely large, the method has a potential to save a lot of memory space. Models whose solution is not practicable due to memory limitations, may be analyzed with this method. During the iteration, different terms of the sums involved can be evaluated independently of each other. This makes it possible to parallelize the scheme.

3. Symmetries

Many modern parallel systems consist of a large number of identical components which operate largely independently of each other but observe the same stochastic rule. The explicit modelling of each individual component often increases the overall number of states in such a way that the model is no longer tractable, i.e. its solution is no longer feasible on today's computer equipment because of the combinatorial state space explosion. However, it is possible to take advantage of the symmetrical properties of such systems as shown in this section.

The example from Section 2 is now continued by adding a third submodel X_3 which is similar to X_2. Figure 2 shows the three submodels represented by their state transition diagrams, two of which are identical. The naïve way of building the generator matrix would be to apply the relation

$$T = (R \oplus S_{X_2} \oplus S_{X_3}) + \left(R_a \otimes S_{a,X_2} \otimes Id_3\right) + \left(R_a \otimes Id_2 \otimes S_{a,X_3}\right)$$

with $S_{X_2} = S_{X_3} = S$ and $S_{a,X_2} = S_{a,X_3} = S_a$, where$R$, S, R_a and S_a are defined as in Section 2. The middle term of the sum represents the dependence between X_2 and X_1, and the

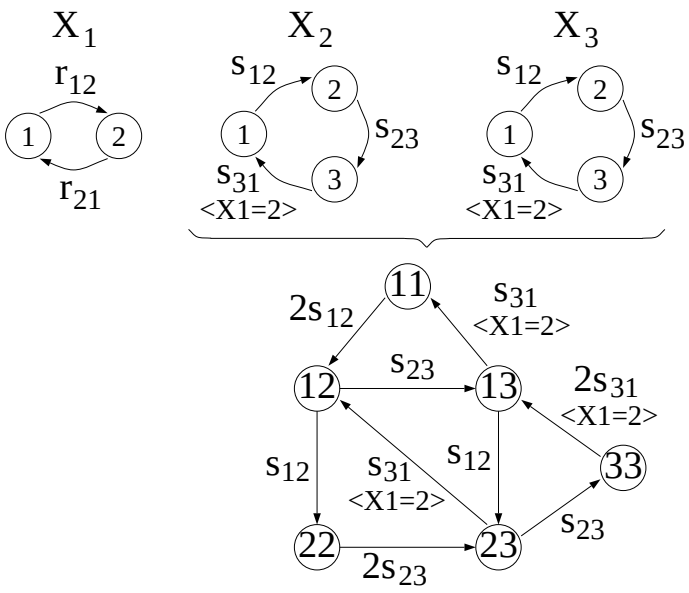

Figure 2: Identical submodels X_2 and X_3 and their combination

third term represents the dependence between X_3 and X_1. This approach would result in an overall number of $2 * 3 * 3 = 18$ states. We note that, from a global point of observation, due to the similarity of X_2 and X_3, there is no need to distinguish between the ordering of the states of submodels X_2 and X_3. For example, states $(1, 1, 2)$ and $(1, 2, 1)$ can be regarded as identical. Representing such pairs of states by a single state results in a reduction of the combined number of states of submodels X_2 and X_3 from $3*3 = 9$ to 6, as illustrated in the lower part of Figure 2. In the reduced model, matrix $S_{X_{23}}$ describes the independent state transitions within the combination of X_2 and X_3, and $S_{a,X_{23}}$ contains the transitions which are dependent on the state of X_1.

$$S_{X_{23}} = \begin{bmatrix} \Sigma & 2s_{12} & & & & & \\ & \Sigma & s_{23} & s_{12} & & & \\ & & \Sigma & & s_{12} & & \\ & & & \Sigma & 2s_{23} & & \\ & & & & \Sigma & s_{23} & \\ & & & & & \Sigma & \end{bmatrix} \qquad S_{a,X_{23}} = \begin{bmatrix} * & & & & & & \\ & * & & & & & \\ s_{31} & & -s_{31} & & & & \\ & & & * & & & \\ & s_{31} & & & -s_{31} & & \\ & & 2s_{31} & & & -2s_{31} & \end{bmatrix}$$

The generator of the overall model now has the form

$$T = (R \oplus S_{X_{23}}) + \left(R_a \otimes S_{a,X_{23}}\right)$$

with only 12 states, as opposed to 18. It is important to note that the method is exact, i.e. the reduction of the state space leads to exactly the same steady-state probabilities and does not involve any approximation.

4. Symmetry Exploitation in Coloured Stochastic Petri Nets

The automation of symmetry detection and exploitation techniques is an important research issue. Presently, no formal approach to the exploitation of symmetries in structured modelling

is known. However, in the field of coloured generalized Petri nets such formal methods have been developed [8]. We suggest the translation of these methods to the domain of structured modelling.

In the context of generalized stochastic Petri nets (GSPNs), identical subnets can be folded, thereby obtaining a concise coloured GSPN representation. Tokens circulating in the coloured Petri net belong to different classes (colours), and places and transitions of the net are associated with colour domains. Classes may be either unstructured or ordered, in which latter case there is a successor-predecessor relationship between tokens of the same class.

Tokens of the same class exhibit a similar behaviour which is one source of symmetry. The permutation of tokens within one class leads to a marking which is stochastically equivalent. Moreover, different colour classes may not have to be distinguished in certain circumstances. This allows the definition of an equivalence relation on the model's state space, such that the set of permutable markings is represented by one symbolic marking (a representative marking is chosen according to lexicographical ordering). This concept yields a symbolic reachability graph (SRG) instead of an ordinary reachability graph. The SRG consists of symbolic markings and transitions between them, thereby reducing the number of model states. The SRG is then transformed into the underlying CTMC whose solution yields the steady-state probabilities of the symbolic markings. The steady-state probability of an ordinary marking is equal to the steady-state probability of the respective symbolic marking divided by the cardinality of its equivalence class.

References

[1] 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.

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

[3] P. Dauphin, F. Hartleb, M. Kienow, V. Mertsiotakis, and A. Quick. PEPP: Performance Evaluation of Parallel Programs — User's Guide – Version 3.1. Technical Report 5/92, Universität Erlangen–Nürnberg, IMMD VII, April 1992.

[4] 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.

[5] M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, February 1981.

[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] S. Donatelli. Superposed Stochastic Automata: a class of Stochastic Petri Nets amenable to parallel solution. In *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models*, pages 54–63, Melbourne, December 1991.

[8] 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.