Structured Markovian Performance Modelling with Automatic Symmetry Exploitation

Markus Siegle

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

1. Introduction

Markov models are widely used for the modelling of parallelism and concurrency because they are amenable to numerical analysis. The major problem in Markovian modelling of complex real systems is the large number of model states. Memory space needed for the infinitesimal 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. The development of efficient techniques for the modelling of parallel systems is therefore of great importance.

Among the techniques which have been considered in order to overcome the state space explosion problem are model decomposition [4], model simplification [9] and structured modelling. In this paper, we present a method and modelling framework following the concept of structured modelling. In particular, we address the state space explosion problem in the context of models which exhibit symmetries.

2. Structured Modelling

Structured modelling is based on information about the modular structure of the real system to be modelled. 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. Among published approaches to structured modelling, the following are of particular relevance:

- Plateau uses stochastic automata for model specification [6, 7]. The numerical solution of the model can be carried out in a *distributed* fashion using a tensor descriptor of Q. This technique saves a lot of memory space but does not reduce the number of computation steps.
- Buchholz employs the tensor algebra approach for hierarchical multi-paradigm models [2].

Our own modelling framework combines and extends features of the above mentioned approaches. While we allow multiple paradigms to be used for submodel specification, we preserve a state-oriented view. 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, which can lead to significant state space reduction (see Sec. 3). Furthermore, with the notion of submodel classes, new and flexible synchronization schemes become possible.

In our modelling framework, the overall model consists of n interacting submodels. Submodels are grouped in submodel classes. There are c submodel classes, class i containing n_i identical (symmetric) copies of submodel i. Therefore, for the total number of submodels we have $n = \Sigma_{i=1}^{c} n_i$. The instantaneous state transition of one or more submodels is called an event. Events are either local, i.e. affecting only one submodel, or synchronizing, to specify the interdependence of submodels. Let E be the set of synchronizing events, and λ_e the rate of the event $e \in E$. Submodel i has s_i states. The internal and external behaviour of submodel i is described by a set of matrices of dimension s_i. Matrix $Q_l^{(i)}$ contains the rates of the local state transitions of submodels from class i. For a synchronizing event $e \in E$, matrix $Q_e^{(i)}$ specifies its influence on the submodels of class i. For a synchronizing event e there are two possible semantics: Either all submodels of a class participate in the event e, or only one submodel per class participates. This feature allows to specify different synchronization patterns very elegantly.

The overall tensor descriptor Q in this modelling framework is given in Eq. (1). The dimension s of Q (the number of states of the overall model) is equal to the product of the dimensions of the submodel matrices, i.e. $s = \Pi_{i=1}^{c} s_i^{n_i}$.

$$Q = \bigoplus_{i=1}^{c} \left(\bigoplus_{j=1}^{n_i} Q_l^{(i)} \right) + \sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} \left(\bigodot_{j=1}^{n_i} Q_e^{(i)} \right) - \bigotimes_{i=1}^{c} \left(\bigodot_{j=1}^{n_i} Q_{e,n}^{(i)} \right) \right) \tag{1}$$

The form of Q illustrates that the generator matrix of the overall model is built from three parts: A part corresponding to the events local to one of the n submodels (built from matrices of type $Q_l^{(i)}$), a second part corresponding to the synchronizing events (matrices $Q_e^{(i)}$) and a third part which is needed to correct the diagonal entries with respect to the synchronizing events (matrices $Q_{e,n}^{(i)}$). If submodels of class i are not affected by a particular synchronizing event e, the corresponding $Q_e^{(i)}$ and $Q_{e,n}^{(i)}$ are identity matrices. In the expression for Q the tensor sum and tensor products are structured in a class-wise fashion. The symbol $\odot$ is interpreted as either $\otimes$ or $\oplus$, depending on the semantics of the event e. If the semantics of e is such that all submodels in class i participate in the synchronization, $\odot$ is replaced by $\otimes$. If only one submodel of class i participates, $\odot$ is replaced by $\oplus$.

The framework just described constitutes a minimal modelling environment which can be extended in several ways. For instance, events may be associated with functional rates, i.e. rates which depend on the current global state of the system. Probabilistic events can be added to ease the modelling of phenomena such as routing probabilities.

We have introduced a structured modelling framework which can fully take advantage of the storage savings of the tensor descriptor technique. The most important enhancement with respect to previous approaches is the notion of submodel class which supports different synchronization patterns and (in the presence of symmetries) leads to a significant state space reduction.

3. Exploiting Symmetries — Submodel Classes and Markov Chain Lumpability

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. Since many modern parallel and distributed systems consist of a high number of similar components (processors, memory modules, stations on a network), and since parallel and distributed application programs often involve replicated processes, models of such systems exhibit symmetries. Therefore symmetry exploitation is an issue of great practical relevance.

The idea of state space reduction through symmetry exploitation is a technique which has been used before. Three recent approaches to a general modelling procedure in which symmetry exploitation is an essential part are the following:

- The concept of symbolic reachability graph (SRG) for coloured stochastic Petri nets [3].
- Stochastic activity networks [8].
- Reduction of hierarchical models [1] in the context of [2].

Our approach to symmetry exploitation is to combine all submodels of a class and replace them by a single reduced model in order to avoid redundant states. During the construction of the reduced model, only one state is chosen as a representative for each set of ‘similar’ states.

Let Z be the state space of the overall model. A state $z \in Z$ is described by the tuple $z = (z_{11}, z_{12}, \ldots z_{1n_1}, z_{21}, z_{22}, \ldots z_{2n_2}, \ldots, z_{c1}, z_{c2}, \ldots z_{cn_c})$. In this tuple, $z_{i,k}$ describes the state of the kth submodel of class i. We consider two states x and y symmetric if y is a permutation of x, i.e. if there exists a permutation matrix P of dimension n such that $y = xP$, with the additional condition that P has a block diagonal structure $P = diag\{P_1, \ldots, P_c\}$. The blocks P_i, in turn, are permutation matrices of dimension n_i. In other words, the permutation by P maps a position onto another position belonging to the same class.

The state space Z is now partitioned in such a way that all symmetric states are in the same set of the partition. This partition is clearly an equivalence relation. Its construction guarantees that the sum of the transition rates from a given state to all states in another set is the same for all states within the same set of the partition. This is exactly the condition for the Markov process with state space Z to be lumpable [5]. A reduced model can then be built which contains only one state per set of symmetric states.

In general, a formal way to compute the generator matrix $\hat{Q}$ of the lumped Markov chain with respect to a partition is to right-multiply the original generator matrix Q with a projection matrix V, and to left-multiply with a selection matrix U.

$$\hat{Q} = UQV \tag{2}$$

Matrix V is used to sum up the columns of Q which correspond to states which are lumpable. In the product QV, all rows corresponding to lumpable states are identical. Matrix U is used to select exactly one such row per set of lumpable states.

A naïve approach to reducing the state space would be to first compute the overall generator matrix Q in its tensor descriptor form given in Eq. (1), and then perform the multiplications with U and V. However, due to the dimension of Q this would be impractical. We will now see that lumping can be carried out at the submodel class level which brings practical advantages. We focus our attention to class i. The combined state space S_i of all submodels of class i has $s_i^{n_i}$ states, where every state is described as an n_i-tuple. Again, two states $x = (x_1, \ldots, x_{n_i})$ and $y = (y_1, \ldots, y_{n_i})$ are symmetric if y is a permutation of x, i.e. if there exists a permutation matrix P_i of dimension n_i such that $y = xP_i$. Note that these are the same P_i as above.

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 $C_{s_i}^{n_i} \times s_i^{n_i}$ and $s_i^{n_i} \times C_{s_i}^{n_i}$, where $C_{s_i}^{n_i}$ is defined as $C_{s_i}^{n_i} = \binom{n_i+s_i-1}{s_i-1}$. Lumping symmetric states in the combined stochastic process of class i reduces the state space of this process from $s_i^{n_i}$ to $C_{s_i}^{n_i}$. We define the following three matrices describing the reduced stochastic process of submodel class i.

$$\hat{Q}_l^{(i)} = U_{s_i}^{n_i} \left(\bigoplus_{j=1}^{n_i} Q_l^{(i)} \right) V_{s_i}^{n_i}, \quad \hat{Q}_e^{(i)} = U_{s_i}^{n_i} \left(\bigodot_{j=1}^{n} Q_e^{(i)} \right) V_{s_i}^{n_i}, \quad \hat{Q}_{e,n}^{(i)} = U_{s_i}^{n_i} \left(\bigodot_{j=1}^{n} Q_{e,n}^{(i)} \right) V_{s_i}^{n_i} \tag{3}$$

Now we will verify that two expressions for the reduced overall Markov chain are the same: The expression we obtain if we apply lumping after building the overall tensor descriptor of Eq. (1), and an expression we obtain if lumping is carried out at the class level.

$$\hat{Q} \overset{!}{=} \bigoplus_{i=1}^{c} \hat{Q}_l^{(i)} + \sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} \hat{Q}_e^{(i)} - \bigotimes_{i=1}^{c} \hat{Q}_{e,n}^{(i)} \right) \tag{4}$$

Due to limited space we only show the equality of the second term. We use the important property $\otimes_{i=1}^{n}(A_i B_i) = (\otimes_{i=1}^{n} A_i)(\otimes_{i=1}^{n} B_i)$ which allows us to write

$$\begin{aligned}\sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} \hat{Q}_e^{(i)} \right) &= \sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} \left(U_{s_i}^{n_i} \left(\bigodot_{j=1}^{n_i} Q_e^{(i)} \right) V_{s_i}^{n_i} \right) \right) = \sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} U_{s_i}^{n_i} \right) \left(\bigotimes_{i=1}^{c} \left(\bigodot_{j=1}^{n_i} Q_e^{(i)} \right) \right) \left(\bigotimes_{i=1}^{c} V_{s_i}^{n_i} \right) \\ &= \sum_{e \in E} \lambda_e U \left(\bigotimes_{i=1}^{c} \left(\bigodot_{j=1}^{n_i} Q_e^{(i)} \right) \right) V = U \left(\sum_{e \in E} \lambda_e \left(\bigotimes_{i=1}^{c} \left(\bigodot_{j=1}^{n_i} Q_e^{(i)} \right) \right) \right) V \end{aligned} \tag{5}$$

Here we have used the relations $U = \otimes_{i=1}^{c} U_{s_i}^{n_i}$ and $V = \otimes_{i=1}^{c} V_{s_i}^{n_i}$ which can be verified by examination of the matrices U, V and $U_{s_i}^{n_i}, V_{s_i}^{n_i}$. For the first and third term of Eq. (4), similar reasoning as in Eq. (5) can be applied.

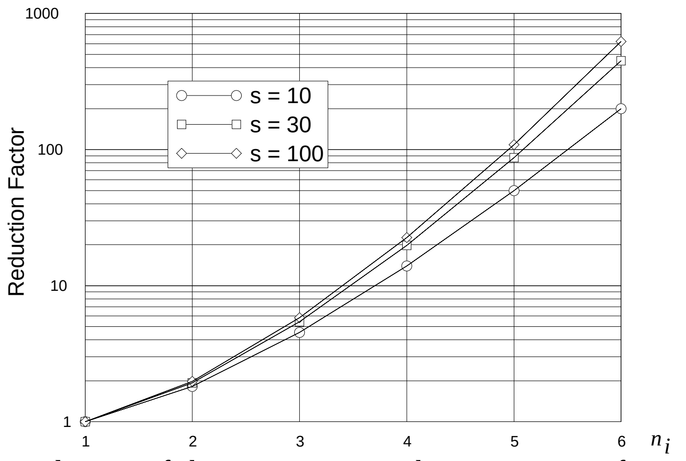

Figure 1: Reduction of the State Space with Respect to n_i for Various s_i

Lumping at the class level has the advantage that the overall generator matrix Q does not have to be built in order to take advantage of symmetries. Instead, the matrices describing the lumped Markov chain are computed independently for each submodel class as a preprocessing step. The matrices involved at the class level are small compared to the dimension of Q. An efficient algorithm for lumping identical submodels will be given in Sec. 4.

In general, the reduction of the state space at the class level is by a factor of $s_i^{n_i}/C_{s_i}^{n_i}$ (The state space of the overall model is reduced by $\Pi_{i=1}^{c} s_i^{n_i}/C_{s_i}^{n_i}$). This factor is charted in Fig. 1 with respect to n_i for submodel classes with varying submodel state space cardinality s_i. It can be observed that the gain of state space reduction is tremendous.

4. Computing the Reduced Generator Efficiently

We now discuss an algorithm to compute matrices of the type $\hat{Q}^{(i)} = U_{s_i}^{n_i}\left(\otimes_{j=1}^{n_i} Q^{(i)}\right)V_{s_i}^{n_i}$ 'on the fly', i.e. directly from the submodel matrix $Q^{(i)}$ without explicitly computing the tensor product inside the parentheses. The idea is to compute only those rows of $\tilde{Q}^{(i)} = \otimes_{j=1}^{n_i} Q^{(i)}$ which will be selected by the left-multiplication with the selector matrix $U_{s_i}^{n_i}$. Within each row, the projection of all symmetric states onto each other (usually achieved by the right-multiplication with the projection matrix $V_{s_i}^{n_i}$) is carried out immediately. In order to be able to do this, the algorithm keeps track of the state description of the originator state $(i_1, \ldots, i_{n_i})$ and of the target state $(j_1, \ldots, j_{n_i})$. Incrementing the state description is done in a special way by the function **increment_ordered()**, which follows the lexicographical ordering but skips those states $(i_1, \ldots, i_{n_i})$ where the condition $i_1 \leq \ldots \leq i_{n_i}$ is violated. Elements of $\tilde{Q}^{(i)}$ are given by

$$\tilde{q}^{(i)}_{(k_1,\ldots,k_{n_i}),(l_1,\ldots,l_{n_i})} = \prod_{j=1}^{n_i} q^{(i)}_{k_j l_j} \tag{6}$$

Therefore the entries of $\hat{Q}^{(i)}$ are computed as

$$\hat{q}^{(i)}_{(k_1,\ldots,k_{n_i}),(l_1,\ldots,l_{n_i})} = \sum_{(m_1,\ldots,m_{n_i})=P(l_1,\ldots,l_{n_i})} \prod_{j=1}^{n_i} q^{(i)}_{k_j m_j} \tag{7}$$

where $P(l_1, \ldots, l_{n_i})$ denotes a permutation of $(l_1, \ldots, l_{n_i})$. A description of the algorithm in pseudo-code is shown at the top of the next page.

For the complexity analysis of this algorithm we use the scheme on the right hand side, which indicates the number of computation steps needed for the respective line. We have used the information that on the average $s_i^{n_i}/C_{s_i}^{n_i}$ permutations have to be considered for a given target state $(j_1, \ldots, j_{n_i})$. The non-reduced generator matrix $\tilde{Q}^{(i)}$ can be computed by an algorithm which has similar structure but uses the ordinary incrementation function and on line (7) employs the expression of Eq. (6) instead of Eq. (7). Comparing the dominant terms of both complexities, it can be observed that the construction of the reduced generator is cheaper than the construction of the non-reduced generator by at least a factor of $s_i^{n_i}/C_{s_i}^{n_i}$. The algorithm to compute $\hat{Q}^{(i)} = U_{s_i}^{n_i}\left(\oplus_{j=1}^{n_i} Q^{(i)}\right)V_{s_i}^{n_i}$ has the same form.

```
(1)   row = 1                                                        1
(2)   $(i_1, \ldots, i_{n_i}) = (1, \ldots, 1)$                      + $n_i$
(3)   while $((i_1, \ldots, i_{n_i}) \neq (s_i, \ldots, s_i))$       + $C^{n_i}_{s_i}$ *
(4)     {col = 1                                                         {1
(5)      $(j_1, \ldots, j_{n_i}) = (1, \ldots, 1)$                       + $n_i$
(6)      while $((j_1, \ldots, j_{n_i}) \neq (s_i, \ldots, s_i))$        + $C^{n_i}_{s_i}$ *
(7)        {$\hat{q}^{(i)}_{row,col} = \hat{q}^{(i)}_{(i_1,\ldots,i_{n_i}),(j_1,\ldots,j_{n_i})}$     {$n_i * s_i^{n_i} / C^{n_i}_{s_i}$
(8)         increment_ordered$((j_1, \ldots, j_{n_i}))$                      + $n_i$
(9)         col++                                                             + 1
(10)       }                                                                 }
(11)     increment_ordered$((i_1, \ldots, i_{n_i}))$                     + $n_i$
(12)     row++                                                           + 1
(13)    }                                                                }
```

The analysis of complexity shows that the computation of $\hat{Q}^{(i)}$ 'on the fly' is already cheaper than the computation of $\tilde{Q}^{(i)}$. This is a very important result. It states that we do not have to pay anything in order to benefit from the reduction of the state space for the subsequent analysis of the Markov chain.

5. Conclusion

In this note, 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, but is extended in such a way that symmetries are recognized and exploited in an automizable way. Our framework is kept general enough to allow multiple paradigms to be employed for the specification of submodels. Exact Markov chain lumpability is the theoretical background for symmetry exploitation, which we show can be 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. Finally, we have presented and analyzed an algorithm for the efficient computation of the generator matrices of the reduced model.

References

[1] P. Buchholz. Hierarchical Markovian Models - Symmetries and Reduction. In R. Pooley and J. Hillston, editors, *6th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 305–319, Edinburgh, September 1992.

[2] P. Buchholz. Numerical Solution Methods Based on Structured Descriptions of Markovian Models. In G. Balbo and G. Serazzi, editors, *Proceedings of the 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 242–258. Elsevier Science Publisher B.V., 1992.

[3] 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.

[4] 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.

[5] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer, 1976.

[6] 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.

[7] 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.

[8] 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.

[9] C. Simone and M. Ajmone-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.