On Efficient Markovian Modelling

Markus Siegle,
Institut für Informatik VII, Universität Erlangen-Nürnberg,
Martensstraße 3, 8520 Erlangen, Germany,
e-mail siegle@informatik.uni-erlangen.de

Abstract

Approaches to the state space explosion problem in Markovian modelling of concurrent systems are discussed. The emphasis is on structured model description techniques and model simplification. In particular, it is shown that the exploitation of model symmetries has a potential to drastically reduce the size of the state space. The common features of different techniques described in the literature are pointed out. Examples from the domain of stochastic Petri Nets are given.

Keywords: Performance Modelling, Continuous Time Markov Chain, State Space Explosion, Stochastic Petri Nets, Symmetry

1 Introduction

High-level description techniques such as stochastic Petri nets, queueing networks or stochastic automata are convenient for modelling the functional behaviour and the performance of parallel and distributed systems. If the time distributions associated with model constructs are exponential, the model's stochastic behaviour is described by an underlying continuous time Markov chain (CTMC). When using such models for modelling complex systems, the combinatorial explosion of the state space is often a serious problem.

This paper discusses techniques which have been developed in order to alleviate, if not overcome, the largeness problem. We present a survey of techniques which have been described in the literature. In particular, we wish to point out the common features of different approaches. We use the term *efficient Markovian modelling* for summarizing these approaches.

Largeness can be approached in a number of different ways: It is possible to deal with largeness at the level of the model description (high-level) or at the level of the underlying CTMC (low-level). The former can be more easily understood by the user because he is working with the high-level representation. Some techniques aim at avoiding largeness during model design, whereas others allow the toleration of largeness during model analysis. Largeness avoidance can be achieved by taking into account knowledge about the structure of the system to be modelled. While some of the approaches are not limited to the modelling paradigm used (e.g. queueing network, GSPN, ...), others are paradigm-specific. Finally, we can distinguish between techniques which yield exact results and others based on approximations. This paper concentrates on the former.

The following section contains an overview of approaches to the largeness problem, based on the literature. All examples in this paper are taken from the domain of stochastic Petri nets. In section 3, the exploitation of model symmetries using Generalized Coloured Stochastic Petri Nets (GCSPNs) is discussed. Section 4 contains simplification techniques for the well-established class of Generalized Stochastic Petri Nets (GSPNs) [14].

2 Approaches to Dealing with Large Markovian Models

As illustrated in fig 1, approaches to overcoming the combinatorial state space explosion problem can be divided into three categories: Model decomposition, structured model description and model simplification.

2.1 Model Decomposition

Starting with a large model, *decomposition* can be applied in order to reduce the expense for solving the underlying Markov chain. Classical results in this category include product form queueing networks [2] and Courtois' notion of *nearly completely decomposable* (NCD) systems [11]. More recently, Ciardo and Trivedi have been working on the decomposition of GSPNs [10], thereby employing the notion of *near-independent subsystems* (NIS). While the decomposition of product form queueing networks is exact, NCD and NIS decomposition typically yield approximate results. The terms NCD and NIS are completely orthogonal, i.e. the fact that a model is NCD does not imply that it is also NIS, and vice versa. Intuitively speaking, in an NCD system, the state space can be partitioned into groups in such a way that transitions between groups are much less likely than transitions within a group. For the decomposition to yield good approximations, entries in the diagonal blocks of the infinitesimal generator matrix of the CTMC have to be significantly larger than entries outside these blocks. A model which is composed of subsystems with almost no mutual interference falls into the class of NIS. The state space of the joined model is equal to the Cartesian product of the subsystems' state spaces. NIS models have a generator matrix structure which is "close" to the tensor sum [12] of the subsystems' generator matrices.

The major difficulty with the decomposition approach is —given the joined model— to recognize the way in which the model should be decomposed. A model is decomposa-

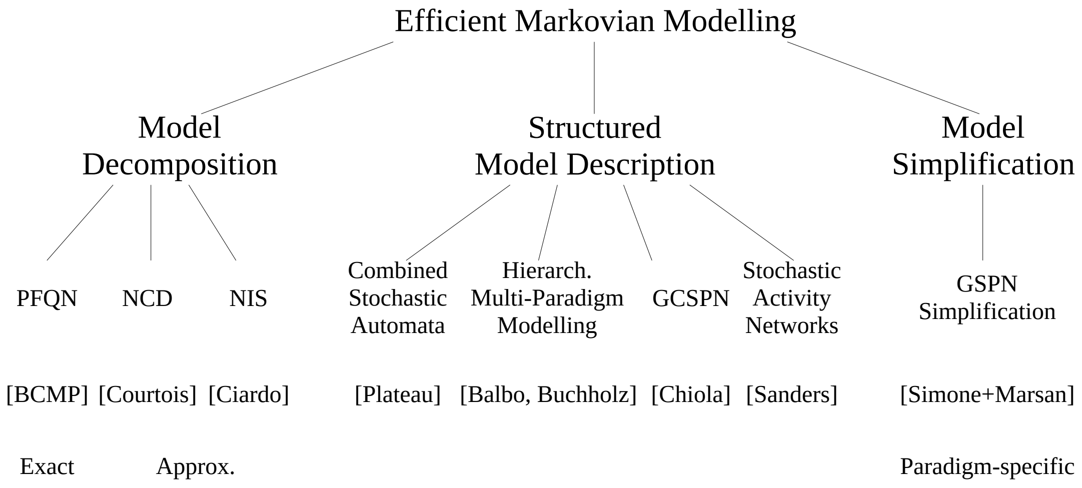
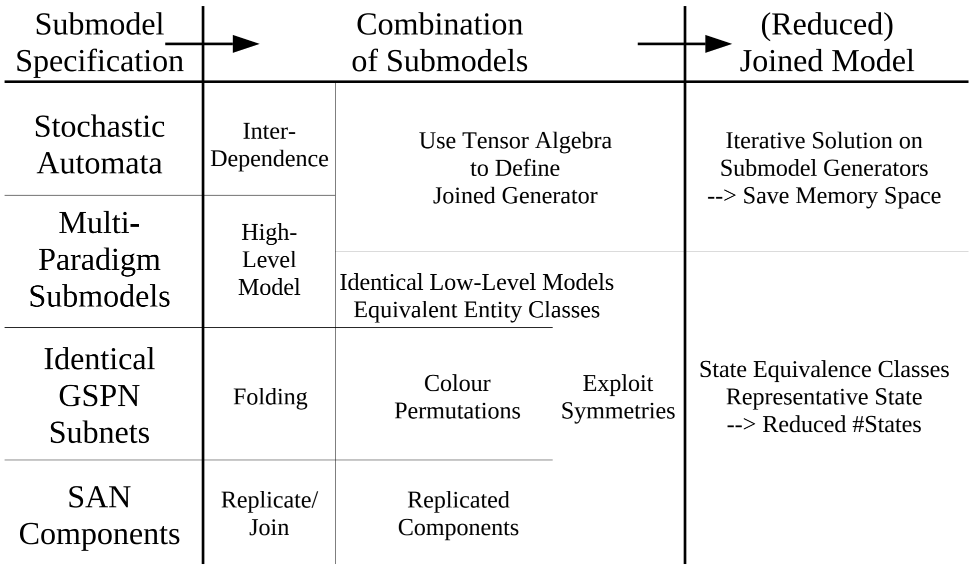


Figure 1: Survey of Techniques for Efficient Markovian Modelling

ble because it is combined of parts which have only limited interaction and interdependence. However, knowledge about the way in which the combination was carried out is not represented in the joined model. Therefore it seems to be more advantageaous to use this knowledge during model design. This leads to structured model description techniques.

2.2 Structured Model Description

There is a large number of approaches which fall into the structured model description category. In [15], Plateau showed how interdependent stochastic automata can be combined to a joined model. Tensor algebra is used to obtain the structure of the combined generator matrix. It is shown that an iterative solution technique (power method) can be applied to solve the combined model, without explicitly generating the joined generator matrix. A similar approach is described by Donatelli [13] for superimposed stochastic automata, a special class of stochastic Petri nets. Balbo et al. described a technique in which queueing networks and GSPNs are combined for solving complex models [1]. A similar approach is followed by Buchholz [4] in the hierarchical multi-paradigm modelling approach. The joined model consists of a number of low level models which can be specified by multi-class queueing networks or coloured stochastic Petri nets, and a high level model which describes the flow of entities (customers or tokens) between the low level models. Buchholz also uses tensor algebra to obtain the combined generator matrix. He shows that beside the power method, other iterative solution techniques (Jacobi and Gauss-Seidel) can be applied based on the submodel generator matrices. In a recent paper, this technique is extended to the exploitation of symmetries in order to reduce the cardinality of the state space [5]. Symmetry exploitation is also the basic idea which led to the use of coloured Petri nets for the purpose of performance evaluation [9]. This will be demonstrated by an example in



Submodel Specification	Combination of Submodels			(Reduced) Joined Model
Stochastic Automata	Inter-Dependence	Use Tensor Algebra to Define Joined Generator		Iterative Solution on Submodel Generators --> Save Memory Space
Multi-Paradigm Submodels	High-Level Model	Identical Low-Level Models Equivalent Entity Classes	Exploit Symmetries	State Equivalence Classes Representative State --> Reduced #States
Identical GSPN Subnets	Folding	Colour Permutations		
SAN Components	Replicate/ Join	Replicated Components		

Figure 2: Structured Model Description: Common Features

section 3. A very similar technique has been described for another class of stochastic Petri nets – stochastic activity networks (SANs) – by Sanders et al. [16]. Here also, the exploitation of symmetries allows to solve the model efficiently.

The common features of different structured model description techniques are pointed out by the table in fig 2. The first column states the technique which is used for submodel specification. The second column is divided into two subcolumns: The technique for combining the submodels is given, and special features which are used during the combination process are mentioned. Properties of the resulting joined model and of its solution method are listed in the third column.

It is common to all approaches that submodels are first specified individually, and then combined in order to obtain the joined model. Different mechanisms are used to specify the way in which the combination takes place. For stochastic automata, the combination is determined by the interdependences which exist between the components. In the hierarchical multi-paradigm modelling approach, the high-level model specifies how low-level models are combined. Using GCSPNs, submodels cannot be identified quite as easily: They can be seen as identical subnets of a GSPN model, where the interaction is given by the arcs and places between them. To build the GCSPN, these identical subnets are folded together, and thereafter only distinguishable by colour domains associated with them. SAN components can be combined using two operations: There is the replicate operation to generate n instances of one component, where a subset of distinguished places is not replicated, i.e. is common to all n instances, and there is the join operation, to join two components by merging two subsets of places, one in each component.

For combined stochastic automata and the hierarchical multi-paradigm modelling approach, the generator matrix of the joined model is given as an expression, in which the generator matrices of the components are combined by tensor operations. Knowledge about the structure of the joined generator matrix is sufficient in order to apply iterative solution techniques for obtaining the steady-state solution of the model, i.e. the joined generator matrix does not have to be created explicitly. Since the joined generator is often a very large sparse matrix, this helps to save a lot of storage space. A second important advantage may be seen by the fact that this kind of iterative solution technique, based on the components' generator matrices, is amenable to parallel processing.

A general expression for the generator matrix, Q, of the joined model is given by

$$Q = \bigoplus_i Q_i + \sum_e \bigotimes_i Q_e^i.$$

In this expression, the tensor sum over the Q_i accounts for state transitions inside one of the submodels. The dimension of Q_i is equal to the cardinalty of the state space of submodel i. Transitions e which affect more than one submodel, such as synchronized events or the instantaneous flow of an entity from one submodel to another, are described by the matrices Q_e^i. Again, the dimension is equal to the cardinalty of submodel i's state space. The summation is over all such transitions, and the tensor product is over all submodels. For those submodels i which are not affected by a particular e, the corresponding Q_e^i is an identity matrix.

For the hierarchical multi-paradigm modelling approach, GCSPN and SAN, it has been shown how the exploitation of model symmetries may reduce the cardinality of the state space dramatically. In the former, symmetries may be present due to a number of identical low-level models, or to the identical behaviour of different entity classes [5]. Entities are either customers (in queueing network submodels) or tokens (in GSPN submodels). In GCSPNs, it is also the identical behaviour of different token classes which enables reduction (see section 3). Here symmetries are recognized by permutating these token classes. Using SANs, symmetries are present wherever the replicate operation is used. Symmetries lead to the partition of the state space into classes of equivalent states, and it suffices to choose one state from each equivalence class —usually determined by lexicographical ordering—to represent this class. The steady-state probability of the representative state in the reduced joined model is equal to the sum of the steady-state probabilities in its equivalence class in the original model.

2.3 Model Simplification

In addition to decomposition and structured model description, the third category of techniques for efficient Markovian modelling is model simplification. A number of approaches to the simplification of GSPN models is presented in section 4. They are based on work by Berthelot dedicated to the structural reduction of place-transition nets [3]. These techniques are applicable only to GSPN models and there is no obvious way to make them available to, say, queueing network models. They are thus paradigm-specific. Similar techniques are discussed in a recent paper by Simone et al. [17].

3 Generalized Coloured Stochastic Petri Nets

In this section, an example is given to show how symmetries can be exploited in GCSPN models.

The analysis of (non-coloured) GSPN models consists of the following steps: Starting from a GSPN which has both timed and immediate transitions, the reachability set is computed to build the reachability graph (RG). Every element of the reachability set corresponds to a possible net marking. The RG contains two classes of markings, tangible markings and vanishing markings. Arcs starting at vanishing markings correspond to the firing of immediate transitions in the net. The next step is the elimination of the vanishing markings, resulting in the tangible RG. The tangible RG may contain redundant arcs and self-loops which have to be eliminated. Then the RG can be transformed into a CTMC.

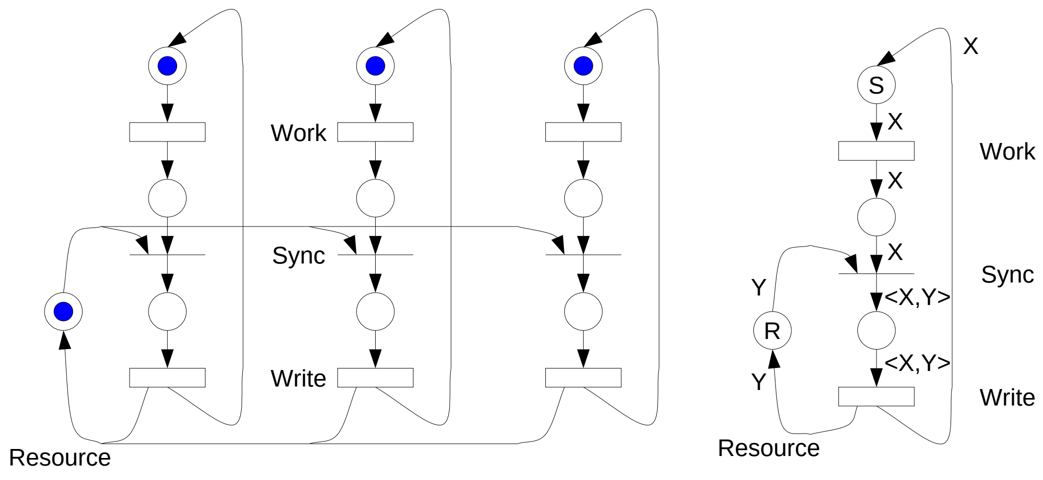


Figure 3: Mutual Exclusion Example

A GSPN model of 3 parallel processes which access a common resource is shown in fig 3 (left). Each process performs the following activities in a cyclic fashion: Having done some work (*work*) it has to wait until the resource is available (*sync*). Then it uses the resource (*write*), returns the resource and goes back to work. The resource can only be used by one process at a time (mutual exclusion), therefore the access to the resource is controlled by the *sync*-transitions.

The scenario shown in fig 3 can be generalized to the case with N processes. In this general case, the total number of reachable markings and thus the cardinality of the reachability set is given by $2^N + N2^{N-1}$. This expression can be obtained by simple combinatorial reasoning: Either none of the processes is currently using the resource, which means that each process is either working or ready to use the resource (2^N different markings), or one of them is using the resource (N different cases) while the others are either working or waiting for the resource (2^{N-1}). It is clear that due to the exponential growth of the reachability set and therefore of the underlying CTMC,

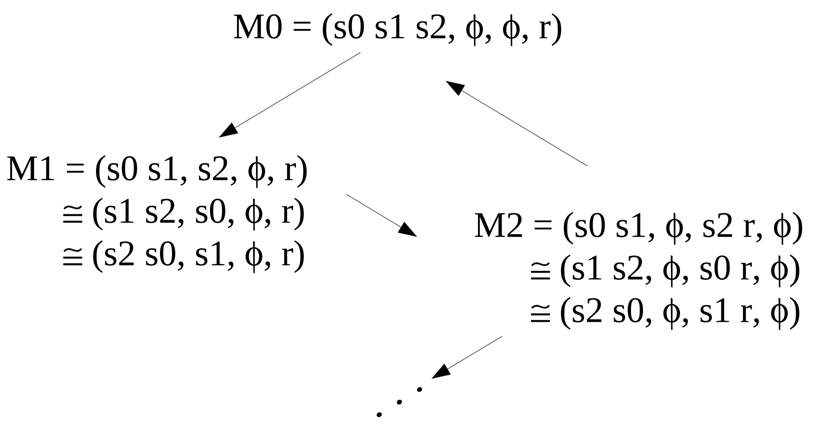


Figure 4: Symbolic Reachability Graph (Part) for the Mutual Exclusion Example

the solution of the model becomes prohibitively expensive even for moderate values of N.

This problem can be overcome by representing the model as a GCSPN and using the concept of *symbolic reachability graph* (SRG). The GCSPN model of the mutual exclusion example is shown in fig 3 (right). To obtain the coloured Petri net, all branches representing the individual processes have been folded onto each other, resulting in a single subnet with the transitions *work, sync* and *write.* The representation of the resource is unchanged. The coloured Petri net is enhanced by annotations containing information about the initial marking and conditions on the transition firing. There are two classes of colours: The elements s_i of class $S = \{s_0, s_1, \ldots, s_{N-1}\}$ represent the N processes, and element r of class $R = \{r\}$ represents the resource. For the initial marking, all processes are working (the input place of transition *work* is marked with S), and the resource is not in use (the *resource* place is marked with R). The arcs are labelled with the functions X and Y which select an element from the sets S and R, respectively. For instance, transition *work* is enabled if there is an element of S in its input place. Upon firing of transition *work*, this element will be moved to the input place of transition *sync.* It is interesting to note what happens at transition *sync.* When it fires, an element of class S is combined with the single element r of class R. The resulting colour is equal to the Cartesian product of S and R, denoted by $< X, Y >$.

There are markings which have identical steady-state probabilities. They can be obtained from each other by permutation of the indices of the elements in class S. For example, if $N = 3$, markings $(s_0 s_1, s_2, \phi, r)$, $(s_1 s_2, s_0, \phi, r)$ and $(s_2 s_0, s_1, \phi, r)$ are interchangeable, because they represent a marking in which two processes are working and the third is ready to use the resource. It is known that these three markings are equally likely, because all three processes behave in exactly the same way, and there is no priority associated with them. The reachability set can therefore be partitioned into classes of equivalent markings. Each equivalence class constitutes a *symbolic marking.*

There is one representative marking in every equivalence class —uniquely determined by lexicographical order—which is used to represent the symbolic marking. The symbolic RG consists of the symbolic markings and transitions between them. Part of the symbolic RG for the GCSPN model in fig 3 is shown in fig 4. It contains the initial marking $M_0 = (s_0 s_1 s_2, \phi, \phi, r)$ and two other symbolic markings, M_1 and M_2. For this mutual exclusion example, the number of symbolic markings and therefore the size of the symbolic RG is given by $2N + 1$. This is also the cardinality of the underlying CTMC which can be seen as a modified version of the original CTMC (obtained from the GSPN model), after equivalent states have been lumped together. In this example, we observe a dramatic reduction of the state space, reducing the exponential growth to linear growth (with respect to N). This demonstrates the power of the symbolic RG method for GCSPNs. For the practical application of this technique, it is of great importance that equivalent markings can be recognized automatically. An algorithm for the computation of the symbolic RG is described in [9]. This result is extended in [7] where the automatic identification of symmetries in the colour structure of a GCSPN is presented.

4 Simplification of GSPNs

In this section, structural simplification techniques for GSPNs are discussed. This work is based on reduction techniques for untimed place-transition nets which were developed by Berthelot [3]. The reduction of place-transition nets has the following goals: checking net properties and the behavioural verification of parallel systems. An initially complex model is reduced in a stepwise fashion until the resulting model is so simple that the desired properties can be easily checked. It is clear that essential net properties, such as boundedness, safety and the covering by invariants, have to be preserved in each reduction step.

For the simplification of GSPNs, the goals have changed: It is the aim of the simplification to make the performance analysis of the model easier. To achieve this, simplifications must reduce the set of reachable markings of a GSPN. More specifically, a reduction of the number of tangible markings is desirable, because the tangible markings correspond to the states of the underlying CTMC. The stochastic behaviour of the model has to be preserved by the simplifications.

Fusion of doubled places. The fusion of doubled places is a reduction technique which can be translated from the place-transition net domain to the GSPN domain without difficulty. Fig 5 (left) shows an example scenario where this kind of simplification is applicable. In this net, tokens in places P_3 and P_4 can never be mixed up, for two reasons: aAt any time, either place P_3 or place P_4 is empty, and any transition which has P_3 as input place has another input place which is not an input place of P_4, and vice versa. Therefore places P_3 and P_4 can be fused into place P_{34}, as shown in the figure (right), without changing the net's behaviour. Application conditions for reduction rules often include conditions on the net's reachability set, i.e. on the net

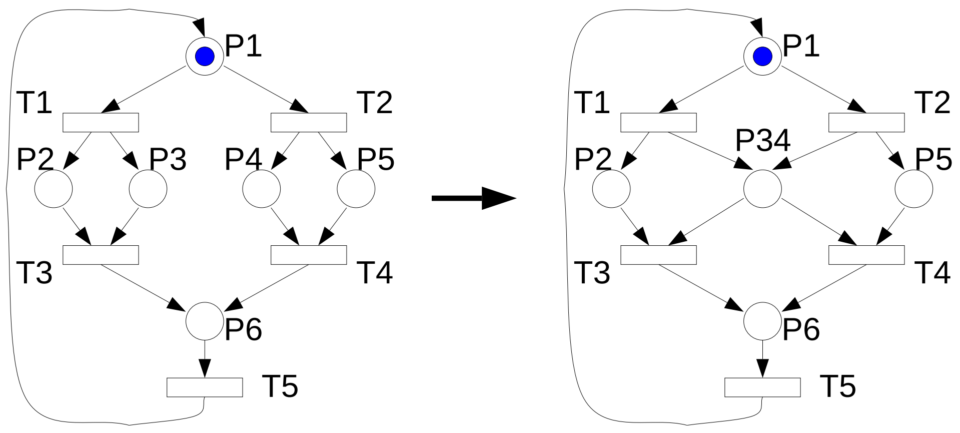


Figure 5: Fusion of Doubled Places

behaviour. Since the computation of the reachability set is expensive, it is desirable to formulate application conditions in terms of conditions on the net structure only. This can be done for the doubled places simplification rule, using the P-invariant $yC = 0$, where C is the incidence matrix of the net. In the example, this suffices to verify that P_3 and P_4 cannot contain tokens concurrently.

We observe that the fusion of doubled places does not contribute to the reduction of the reachability set. The simplified net's reachability set contains the same number of markings as that of the original net. With this simplification, we have only obtained a simpler graphical representation of the net. The same negative result holds for another reduction rule, the elimination of redundant places [3], often also called implicit places.

Pre-fusion of transitions. Fig 6 shows how another reduction technique for place-transition nets can be made available to the simplification of GSPNs. Immediate

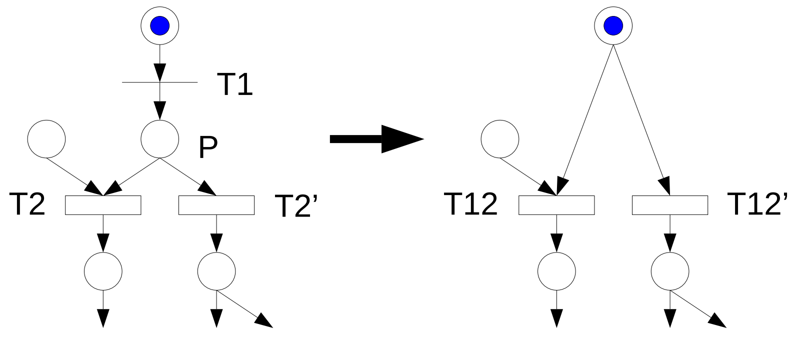


Figure 6: Pre-fusion of Transitions

transition T_1 can be fused with both of the two timed transitions T_2 and T_2'. This is similar to the elimination of immediate transitions at the net level (instead of eliminating vanishing markings at the RG level), as described in [8]. For a net with a large number of immediate transitions, this simplification has the potential to save a lot of memory space during the construction of the underlying CTMC, because no memory space is needed for those vanishing markings which are caused by the now eliminated immediate transitions. However, the number of tangible markings in the reachability set and therefore the size of the CTMC is not reduced by this simplification.

Exchanging conflicts of immediate/timed transitions. Now we will discuss a simplification technique for GSPNs which is capable of reducing the size of the underlying CTMC. An example net which can be simplified by this rule is shown in fig 7 (top). Two levels of decision are represented by the net. At the first level, there

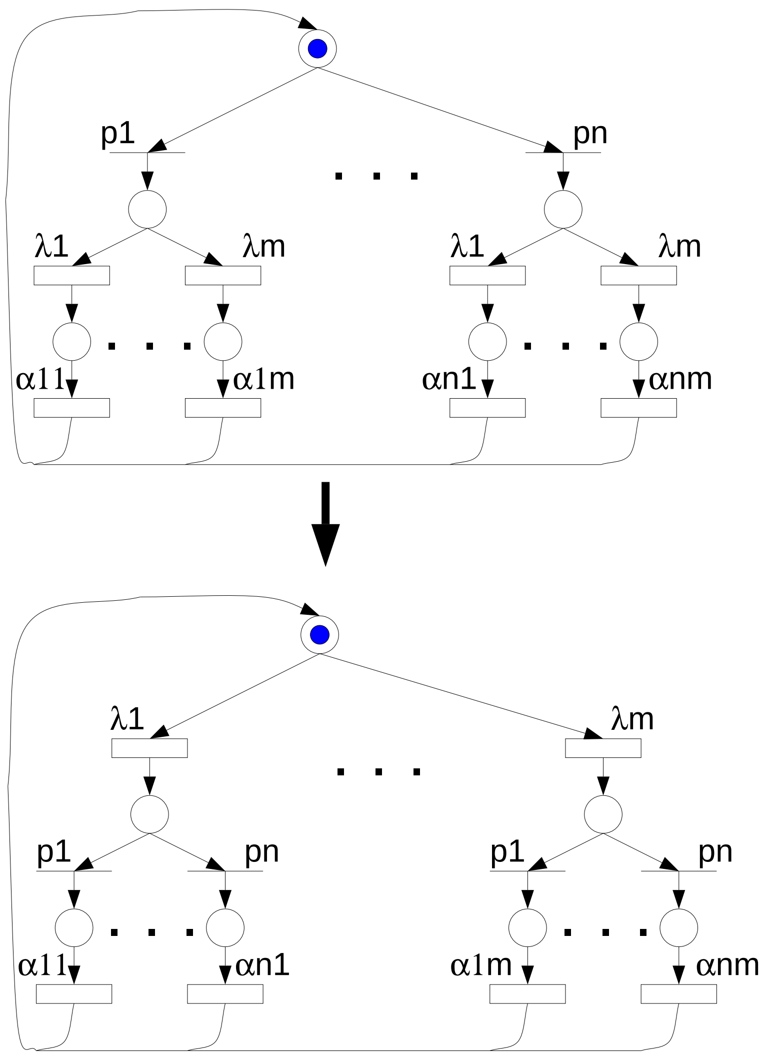


Figure 7: Exchanging Conflicts of Immediate/Timed Transitions

is a conflict between n immediate transitions which is decided according to the firing probabilities $p_1 \dots p_n$ assigned to these transitions. At the second level, in each of the n branches of the net there is a conflict of m timed transitions. This race condition is decided upon the firing rates $\lambda_1 \dots \lambda_m$. The number m of timed transitions has to be the same in each of the n branches, and the set of rates $\lambda_1 \dots \lambda_m$ assigned to these transitions must also be identical in all n branches. The transitions associated with the firing rates $\alpha_{11} \dots \alpha_{nm}$ may be generalized: They may be replaced by arbitrary subnets.

In the simplified net shown in fig 7 (bottom), the two levels of decision are exchanged. The conflict of the m timed transitions is now followed by a conflict of the n immediate transitions in each of the m branches. The rates of the timed transitions labelled with α_{ij} (the subnets in the general case) have to be permutated as shown in the figure: the combination of an immediate transition and a timed transition must lead to the same α_{ij} in the original and in the simplified net. The number of tangible markings for the original net is given by $n + mn$, whereas the simplified net has only $1 + mn$ tangible markings. There is one tangible marking corresponding to the conflict of the m timed transitions in each of the n branches of the original net. In the simplified net, one marking suffices to represent the conflict between the m timed transitions, before a decision is made which of the n immediate transition will fire.

Benefit of the Simplification. Three simplification rules have been presented. It can be observed from the three examples that simplification rules for GSPNs may be classified as follows: There are those which do not change the net's reachability set (they merely change the graphical representation of the net), those which reduce the number of vanishing markings (which saves memory space during the construction of the CTMC) and those which reduce the number of tangible markings of the net (resulting in a smaller CTMC). There is a number of important questions which remains open in the context of GSPN simplification:

- The cost of the simplification needs to be determined. The time spent on the searching for net constructs which may be simplified, on the checking of application conditions and on the execution of simplification rules, plus the time needed to solve the simplified GSPN, must be compared with the time needed to solve the original GSPN.
- It is not known to what extent GSPNs from practical modelling contain constructs amenable to simplification.
- An important point is the semantics of the simplified net. Net constructs such as places and transitions represent constructs from the real world system being modelled, such as processors, buffers or communication channels. If net constructs are manipulated by simplification rules, it is important to relate the semantics of the simplified net to the real world system.
- The examples discussed in this section yield an exact simplification of GSPNs. However, simplifications leading to an approximation of the net's stochastic behaviour may be acceptable. The derivation of lower bounds for the throughput of Markovian Petri nets by means of transformation techniques is described in [6].

- The extension of GSPN approximation rules to GCSPN has not been formalized yet.

Finally, the author is not aware of any tools which support the simplification of GSPNs. Only with the help of tools will simplifications be widely applicable in practical modelling.

5 Summary

We have discussed techniques for making the Markovian modelling of complex systems feasible by reducing the cardinality of the state space. The largeness problem is approached at the high-level model description, rather than at the continuous time Markov chain level. This makes it easier for the user to understand the techniques. We have shown that different techniques for structured model description rely on the same foundations: they use tensor algebra for the description of the joined model's stochastic generator matrix, and they provide means to exploit symmetries in the model, thereby reducing the state space without changing the stochastic behaviour. Transformation techniques for the structural simplification of GSPNs have been presented. They may be used in conjunction with the above mentioned structured modelling techniques by simplifying individual submodels before analyzing the joined model.

References

[1] G. Balbo, S.C. Bruell, and S.Ghanta. Combining Queueing Networks and Generalized Stochastic Petri Nets for the Solution of Complex Models of System Behaviour. *IEEE Transactions on Computers*, 37(10):1251–1268, Oct. 1988.

[2] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2):248–260, 1975.

[3] G. Berthelot. Transformations and Decompositions of Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1986, LNCS 254*, pages 359–376. Springer, 1987.

[4] 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.

[5] 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.

[6] J. Campos, B. Sanchez, and M. Silva. Throughput Lower Bounds for Markovian Petri Nets: Transformation Techniques. In *4th Intern. Workshop on Petri Nets and Performance Models*, Melbourne, Australia, Dec. 1991.

[7] G. Chiola, , and G. Franceschinis. A Structural Color Simplification in Well-Formed Colored Nets. In *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*, pages 144–153, Melbourne, December 1991. IEEE.

[8] G. Chiola, S. Donatelli, and G. Franceschinis. GSPNs versus SPNs: what is the actual role of immediate transitions? In *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*, pages 20–31, Melbourne, December 1991. IEEE.

[9] 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.

[10] G. Ciardo and K.S. Trivedi. A Decomposition Approach for Stochastic Petri Net Models. In *Proceedings of the Fourth International Workshop on Petri nets and Performance Models*, pages 74–83, Melbourne, December 1991.

[11] P.J. Courtois. *Decomposability, queueing and computer system applications.* ACM monograph series, 1977.

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

[13] 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.

[14] M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 1984.

[15] 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.

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

[17] 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.