QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets

Falko Bause, Peter Buchholz, Peter Kemper

University of Dortmund, Informatik IV, D-44221 Dortmund, Germany
e-mail: {bause,buchholz,kemper}@ls4.informatik.uni-dortmund.de

Abstract. This article describes a new version of the QPN-Tool now supporting specification and analysis of hierarchically combined Queueing Petri nets (HQPNs). HQPNs are an extension of QPNs allowing the refinement of places by QPN subnets and/or queues. HQPNs can be analysed with respect to qualitative and quantitative aspects. Quantitative analysis is based on numerical Markov chain analysis. In contrast to conventional techniques the Markov chain underlying a HQPN is analysed by an approach exploiting the hierarchical structure of the model which results in a tensor representation of the generator matrix. This technique extends the size of solvable state spaces by one order of magnitude. Qualitative analysis of HQPNs relies on efficient analysis techniques based on Petri net theory. The new version of QPN-Tool implements the above analysis approaches supported by a graphical interface for a convenient specification of complex models.

Keywords: Hierarchical Modelling, Queueing Networks, Coloured GSPNs, Tensor Based Numerical Analysis, Combined Qualitative and Quantitative Analysis

1 Introduction

The model based analysis of computer and communication systems requires adequate software tools to handle the inherent complexity of today's systems. Furthermore it is more and more agreed upon that analysis has to be performed according to both qualitative and quantitative aspects of a system, preferably using one model. Many modelling paradigms and analysis techniques exist for these purposes, in particular models based on underlying Markov chains are well suited for a quantitative analysis and can also be used for qualitative analysis by neglecting timing information. Markov chains are often analysed by numerical techniques, which are more accurate than simulation and more flexible than analytical analysis techniques. However, any approach for a state based analysis is faced with the following problems:

It has to support the modelling of complex systems, such that models are understandable and can be easily modified, **and** it has to manage the state space explosion which results from the usually exponential growth of the number of states as a function of the size of the model specification.

A rich variety of techniques exists for a high level specification of Markov chains; two particularly important ones concerning performance analysis are extended queueing networks and generalised stochastic Petri nets (GSPNs, [1]).

Both techniques have their specific advantages and disadvantages. The possible exploitation of the advantages of both approaches led to the development of QPNs, combining coloured GSPNs and queueing networks in one modelling formalism [2], and the QPN-Tool supporting the graphical specification and automatic analysis of QPNs [4]. However, QPNs also suffer from the above mentioned problems since they describe a flat model which becomes complex when the system to be modelled is complex and the size of the resulting Markov chain often exceeds the capacity of today's computers.

A general method to handle complexity is the use of hierarchical structuring mechanisms. This route has already been pursued in some performance modelling tools, as an example we refer to HIT [6]. Furthermore hierarchical untimed Petri net models are widespread (e.g. [16]). However, all mentioned approaches use hierarchies only for specification purposes, while analysis is performed on the underlying flat model. More recently a hierarchical approach for the specification and analysis of queueing networks [9], coloured GSPNs [7] and a subclass of HQPNs [3] has been developed. This approach allows the hierarchical specification of models and an efficient numerical analysis that exploits the hierarchical structure. The central idea of the analysis is to represent the huge generator matrix of the underlying Markov chain by much smaller matrices, each describing a submodel, which are combined using tensor operations. Using this technique, Markov chains can be solved which are about an order of magnitude larger than those analyzable with conventional techniques. A salient property of our hierarchical approach is that it does not depend on model symmetries and leads to exact results.

HQPNs as introduced here allow the specification of a system in several levels: The top-level is a QPN additionally comprising special places which themselves include HQPN subnets. The lowest levels of the hierarchy are described by QPN subnets which are coloured GSPNs with some additional places including queues. In this paper we describe a new version of the QPN-Tool [4] which supports the specification and analysis of HQPNs. Specification of HQPNs is supported by a graphical interface which allows a convenient specification and the reuse of submodel specifications in one or several models. Analysis can be performed according to qualitative and quantitative aspects using the same model. Qualitative analysis is based on established algorithms for the analysis of Petri nets. QPN-Tool offers standard Petri net algorithms and also very efficient methods based on special net classes [17]. Quantitative analysis can be performed using conventional analysis techniques on the flat Markov chain, which is often preferable for small state spaces up to 50,000 states, or by the hierarchical technique, which allows the analysis of models with several millions of states on standard workstations. For larger models, approximate analysis based on aggregation/disaggregation can be performed.

A variety of other tools have been developed in the last years to support the specification and analysis of stochastically time-augmented Petri nets, e.g.: DSPNexpress [19], GreatSPN [11], SPNP [12], TimeNET [15] and UltraSAN [23]. All these tools provide numerical solution facilities for the underlying Markov chain, some also include simulation. Apart from UltraSAN all tools rely on flat and uncoloured nets. As far as the authors know there exists presently one other tool which supports Markov chain analysis based on a tensor representation of the generator matrix, namely the tool PEPS for the analysis of stochastic automata networks (SANs) [21]. However, SANs are completely different from the model class introduced here since they specify stochastic automata which communicate via synchronised transitions, rather than by asynchronous exchange of entities. There is also work going on to transfer the results for SANs in the GSPN area [14].

The outline of the rest of this paper is as follows. In the next section we introduce HQPNs in some more detail and briefly explain the hierarchical analysis approach. The following section is dedicated to the QPN-Tool version 2.0. In Sect. 4 a simple example model is presented to compare the performance of different analysis approaches.

2 Hierarchically Combined Queueing Petri Nets

In the original QPN formalism [2] the timing aspects of a coloured GSPN [13] are extended by the option of integrating queues into places (see Fig. 1). Such a timed place consists of a queue and a depository for served tokens. Tokens being added to a timed place, after a transition firing, are inserted into the queue according to the queue's scheduling strategy. Each colour of tokens has an associated individual service time distribution of Coxian type. After service the corresponding token moves to a depository, where it is available to the place's output transitions. Tokens in a queue are not available for firing.

Every QPN can be analysed with respect to qualitative and quantitative aspects. The qualitative analysis of a QPN is completely based on the QPN's underlying coloured Petri net, where all timing aspects are neglected. The basis for a quantitative analysis of a QPN is its corresponding Markov chain. Every QPN describes a stochastic process. A state of this process is determined by the cartesian product of the state descriptors of all timed places and the number of coloured tokens within ordinary places. The initial marking of the QPN defines the initial state of its stochastic process under the assumption that initially all tokens contained in timed places are situated on the corresponding depository. The state space is partitioned into two types of states similar to GSPNs [1]: vanishing and tangible. A quantitative analysis can be performed with conventional techniques [2]. The complexity of a quantitative analysis is significantly reduced if the QPN has a hierarchical structure.

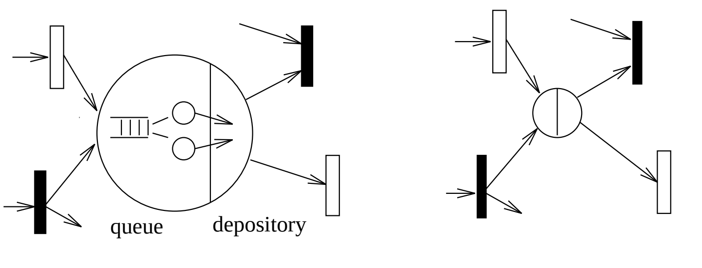


Fig. 1. Example timed place of a QPN and its shorthand notation

Hierarchically combined Queueing Petri nets (HQPNs) are obtained by a natural generalisation of the original QPN formalism. In HQPNs a timed place may contain a whole HQPN subnet instead of a single queue (cf. Fig. 2). Such a place is called a *subnet place.* A HQPN subnet might be a QPN subnet or again a HQPN comprising additional subnet places.

The HQPN subnet of a subnet place has a dedicated input and output place, which are ordinary places of a coloured Petri net. Tokens being inserted into a subnet place after a transition firing are actually added to the input place of the corresponding HQPN subnet. The semantics of the output place of a subnet place is similar to the semantics of the depository of a timed place: Tokens contained in the output place are available for the output transitions of the subnet place; tokens contained in all other places of the HQPN subnet are not available for the output transitions of the subnet place. The HQPN subnets we consider in this context have to observe the conditions given in the description of the hierarchical analysis approach. HQPNs can be analysed as flat models with respect to qualitative and quantitative aspects employing well-known algorithms of Petri net and Markov theory. In the sequel we concentrate on the new approach for quantitative analysis.

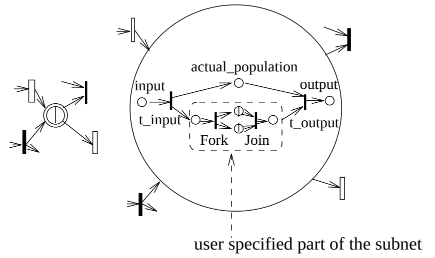


Fig. 2. Example subnet place of a HQPN and its shorthand representation

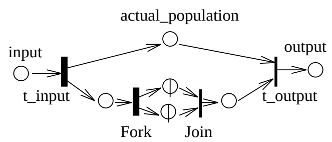


Fig. 3. Isolated HQPN subnet

Integration of subnet places into the original QPN formalism leads to a hierarchical structure of a QPN model. Such a hierarchical structure can surely contain more than just two levels but for simplicity of notation we restrict ourselves in the following to a two-level hierarchy. Let *high-level QPN (HLQPN)* denote the upper level of a HQPN. The HQPN subnets of the subnet places in the HLQPN are denoted by *low-level QPNs (LLQPNs).*

The quantitative analysis employing this hierarchical structure, called *structured analysis,* comprises the following steps:

1. Generation of the HLQPN state space and transition matrix. All subnet places of the HLQPN are replaced by the subnet's dedicated input and output places. Each input and corresponding output place is connected via a timed transition, which behaves like the output transition of the LLQPN and has a nonzero transition rate. We will call such a timed transition *virtual* since the firing rate might be chosen arbitrarily. The initial marking of the HLQPN is given by the initial marking specified on that level. The specification of the HLQPN is independent of the internals of the LLQPNs apart from their output transitions, which determine the input/output behaviour of the LLQPNs. Provided the HLQPN includes no timeless traps, which can be checked on the net level [2], we can determine its state space and transition matrix containing only tangible states. LLQPN specifications have to observe the restrictions given below.

2. Generation of the LLQPNs' state spaces and transition matrices. For a structured analysis, the subnet integrated into a subnet place is restricted to HQPN subnets satisfying the following restrictions [7]:

1. The set of token colours are identical for the input and the output place. Arcs are not allowed which connect the dedicated output place with a transition of the subnet.
2. The macro state descriptor of the LLQPN, as needed for step 1, is given by a fixed set of places. In the current version of the QPN-Tool this set is determined by the places *input* and *actual_population*. Transition *t_input* is defined to behave like an identity function, so that transition *t_output* describes the behaviour of the HQPN subnet as seen in the HLQPN.
3. The subnet is structurally bounded, i.e. for any number n of tokens entering the subnet, the number of all tokens in the subnet is always bounded above by some value $k_n \in \mathbb{N}$. The modeller can check this restriction employing a local qualitative analysis of the isolated subnet (cf. Sec. 3), e.g. calculating a cover of positive P-invariants.
4. According to our assumptions about the influence of the HLQPN on the behaviour of a subnet we restrict this behaviour at the LLQPN level to a kind of single step behaviour: If a token enters the subnet (being fired onto the input place), no token can be fired onto the output place at the same (!) point of time.

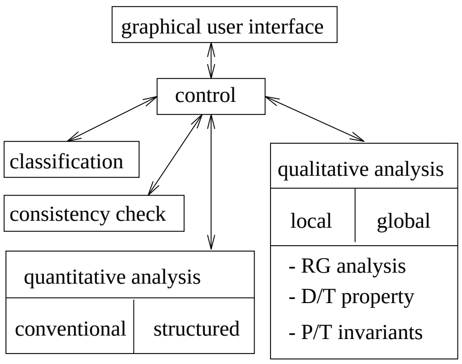
Given these restrictions the state space and transition matrix for each LLQPN are generated in isolation (see Fig. 3). The behaviour of a LLQPN's environment is completely specified by the state transitions of the HLQPN determined in step 1.

In the quantitative analysis timed places can be viewed as special cases of subnet places where the queue forms the HQPN subnet and the output place is given by the depository of the timed place. A timed place trivially satisfies the above given restrictions. It can consequently be handled as a subnet place where the queue and the corresponding depository are considered as a LLQPN.

3 QPN-Tool version 2.0

The QPN-Tool as described in [4] has been modified to allow the specification of HQPNs and their analysis using the efficient technique mentioned in the former section. In the following we describe features of QPN-Tool emphasizing the newly integrated ones.

QPN-Tool is implemented in C and is executable on Sparc-machines with Sunview or OpenWindows under Solaris 1. It contains a graphical user interface and a variety of analysis algorithms for qualitative and quantitative analysis of HQPNs. Figure 4 describes the modular structure of QPN-Tool.



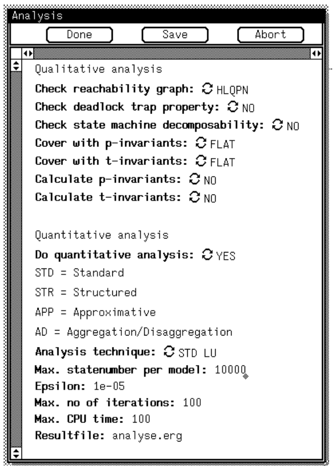


Fig. 4. Modular structure of QPN-Tool

Fig. 5. QPN-Tool's analysis algorithms

3.1 Graphical user interface

The graphical user interface manages the complete user interaction. It supports specification of HQPN models and analysis tasks. It offers a selection of analysis algorithms and performance measures to be calculated and presents results of qualitative and quantitative analysis.

A hierarchical model description consists of two different types of nets: a HLQPN and LLQPNs, which may or may not contain LLQPNs themselves. We give a brief description of a hierarchical model specification process starting with the HLQPN:

1. The graphical description is clearly dominated by its coloured Petri net part. This part is specified by creating and positioning places and transitions and establishing their connections by directed arcs. Figure 6 shows an example HLQPN. Here all transitions are immediate and all places are subnet places except for the place *Manager* which is a timed place.

2. Attributes of places and transitions have to be set. This includes entering the different colours in a list, choosing the type of transitions and places (timed or immediate, resp. timed, subnet or ordinary) and fixing the number of tokens for the initial marking. Timed and subnet places require additional information. For these places, the user has to specify performance figures to be determined in quantitative analysis (cf. Sect. 3.2). In case of a timed place, the scheduling strategy and number of servers must be chosen. Presently available scheduling strategies are Random, PS, Infinite Server (IS) and Priorities. For each colour, the service time distribution is specified by its mean and coefficient of variation, which is then approximated by a Cox distribution. In case of a subnet place, the corresponding subnet specification (see **4.** below) has to be given.

3. The incidence functions of the coloured Petri net (cf. [13]) have to be specified. This is possible at each transition of a graphical submodel. Such a submodel describes the locally unfolded net regarding a single transition and its input and output places. A special feature of QPN-Tool fills the submodel automatically with the transition's colours and all input and output places including all of their colours. Thus only arcs and their weights have to be specified manually. This kind of specification remains unchanged from version 1.0, see [4] for more details.

4. The specification of subnet places follows the same steps as the specification for the HLQPN. Similar to the local unfolding of a transition QPN-Tool automatically fills the subnet with the dedicated input and output places and input and output transitions (Fig. 7, see also *t_input* and *t_output* in Fig. 2). The ordinary place *actual_population* is also inserted automatically.

The automatically generated subnet can be extended by the user to a complete HQPN itself (cf. the user specified part in Fig. 2). The corresponding specification mechanisms are identical to the ones of the former described HLQPN.

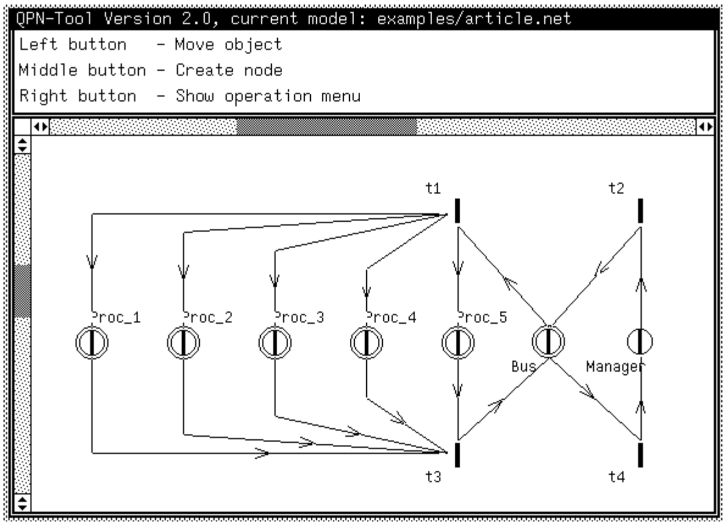


Fig. 6. Example HLQPN

In order to simplify the utilisation of a subnet specification for different places, QPN-Tool allows to declare an ‘external’ subnet by a filename. If the subnet contained in this file has input/output places with colour sets different from the colour set of the subnet place, the colours are matched by order of appearance in the colour lists. This matching is brute-force, but from a user's point of view it is easy and convenient to use. Figure 7 shows a subnet of *Proc_1* which is re-used for subnet places *Proc_2*, ..., *Proc_5* in Fig. 6.

3.2 Analysis techniques of QPN-Tool

Typically analysis goals either refer to qualitative properties, e.g. absence of deadlocks, liveness or boundedness, or to the determination of performance measures. Within QPN-Tool qualitative properties are investigated by a so-called qualitative analysis which is based on Petri net theory. Performance measures, on the other hand, are computed by analysis of the corresponding Markov chain, which is called quantitative analysis. A major advantage of a hierarchical model description is that isolated LLQPNs can be analysed assuming a well-behaving

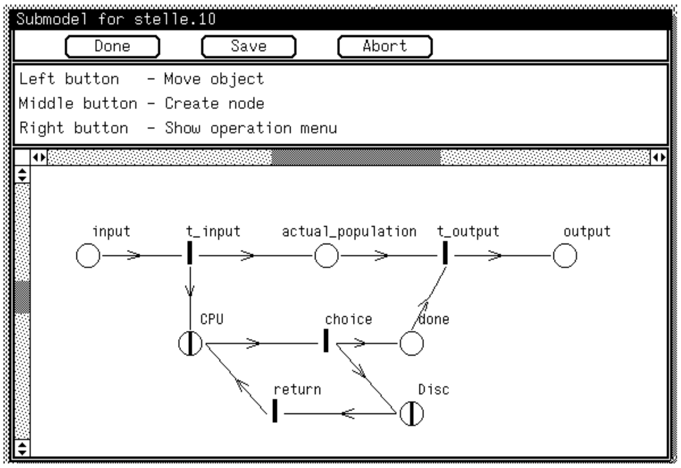


Fig. 7. Subnet of places *Proc_1, ..., Proc_5*

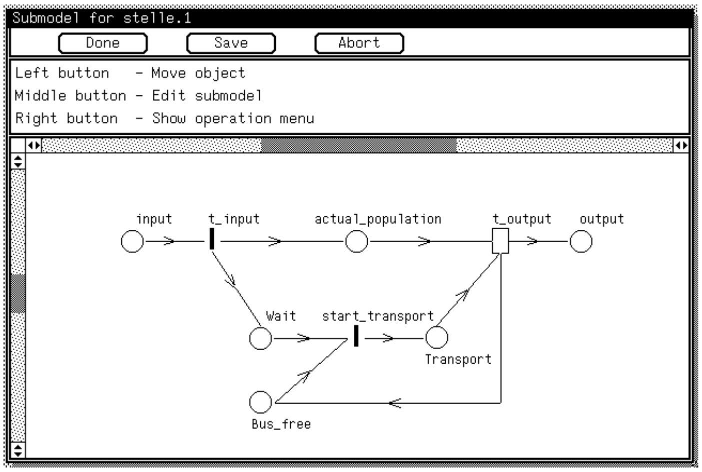


Fig. 8. Subnet of place *Bus*

environment and that the HLQPN can be analysed assuming well-behaving LLQPNs.

Ahead of any qualitative analysis a consistency check is performed, where naming inconsistencies and mismatches between different levels of a hierarchical model are recognized. Subsequent to the consistency check, the embedded CPN of an isolated subnet or the HLQPN itself is unfolded and classified in terms of an uncoloured Place/Transition net: marked graph, state machine, free choice, etc. This classification neglects timing information, replaces a subnet place by its input and output place and connects these places via a virtual transition. It supports the choice of a suitable analysis algorithm for a local qualitative analysis, as special algorithms are available for certain net classes, see Fig. 5.

Qualitative analysis For a qualitative analysis of a HQPN, it can be transformed to a flat CPN neglecting all timing aspects and inserting all subnets for their corresponding subnet places. Figure 5 presents the selection of qualitative analysis algorithms implemented in QPN-Tool; it follows mainly version 1.0, [4]. Apart from 'classical' algorithms like reachability graph analysis and calculation of P- and T-invariants, novel algorithms for special net classes are offered. Qualitative analysis within QPN-Tool aims at liveness and boundedness. If the HQPN does not exhibit these properties, information from the employed algorithm is extracted in order to indicate the reason for a HQPN being unbounded or not live. All implemented qualitative analysis algorithms are based on Petri net theory and consider the underlying untimed CPN, i.e., timing aspects such as firing delays and frequencies are ignored, and a timed place is replaced by two places and a transition connecting these places.

The same set of Petri net analysis algorithms can be applied to the HLQPN. In this case every subnet place is replaced by an input and output place and an additional transition obtained from t_output in the corresponding subnet. For isolated subnets only invariant analysis can be performed.

Conventional and structured quantitative analysis Quantitative Analysis is pursued with the objective of assessing performance properties for a HQPN. Different performance measures are offered for

ordinary places: token population
timed and subnet places: utilisation, throughput and token population

The calculation of performance measures results in mean values for steady state. For token populations variance and distribution are additionally calculated. Results can be computed for all colours of a place separately or aggregated overall colours.

QPN-Tool offers in its current version two main options for a quantitative analysis: conventional and structured. The conventional analysis technique transforms the HQPN into a flat QPN model, maps this QPN model onto a corresponding Markov chain, and subsequently analyses this chain with respect to its steady state distribution. The QPN's state space is fully explored causing quantitative analysis to be restricted to QPNs with a finite state space of acceptable size.

The conventional numerical analysis includes three main steps:

1. exploring the state space
2. computing the steady state distribution
3. calculating the performance measures

It is able to handle state spaces with around $200,000$ states depending on the structure of the CTMC and the available memory. Different algorithms for the calculation of the steady state distribution are offered.

The idea of the structured analysis is to avoid the generation and storage of the complete generator matrix and instead exploit the model structure also for analysis. The underlying ideas are described in [7, 9]. Here we give only a brief summary of the approach. In a first step the HQPN is transformed into a two level hierarchy including one HLQPN and several LLQPNs. Afterwards a single state in the complete state space can be represented by a unique state of the HLQPN and a unique state for every LLQPN. Therefore the complete state space can be generated by combining subspaces of the LLQPN state spaces according to a state of the HLQPN. Subspaces of LLQPN state spaces are defined by the marking of the places *input* and *actual_population*. Like the state space, the generator matrix is structured into submatrices according to the state of the HLQPN and every submatrix can be represented by the generalised tensor (kronecker) product/sum of submatrices describing local transitions in the LLQPNs. These structured representations of state space and generator matrix are very compact since the size of the overall state space and also the size of the generator matrix grows with the product of the sizes of LLQPN state spaces and matrices. Thus, the combination of a few LLQPNs, with a few hundred states each, yields an overall state space with several millions of states.

Structured analysis results from the observation that the matrix structure can be exploited during the analysis. Exploitation of this structure means that state space generation and iterative solution becomes more efficient in terms of memory requirements and sometimes also in terms of CPU time requirements. The underlying numerical techniques are essentially the same as in the conventional case, only the basic operation, the vector matrix multiplication, is implemented in a different form. Thus, structured analysis is an exact approach, as far as iterative numerical analysis techniques are exact, and does only rely on the hierarchical structure, not on symmetry assumptions. For structured analysis we have to perform the same three steps as above, however, the single steps are slightly different. Exploration of the state space consists of

1.1 exploration of the HLQPN state space and elimination of vanishing states. This yields all possible populations (macro states) for all LLQPNs and all queues in the queueing places. Additionally all possible arrivals to LLQPNs are described.
1.2 exploration of the LLQPN state spaces and elimination of vanishing states. The state space and the transition matrix are generated for the LLQPN in combination with an environment generating the same inputs as the HLQPN, but without quantitative information about possible interarrival times.

This approach accelerates generation of the state space for large models, since a number of small state spaces instead of one very large state space is generated. Additionally the generation can be easily parallelised since state space and matrix generation for a LLQPN depend only on the state space of the HLQPN and not on other LLQPNs.

Subsequently the second step is performed and the Markov chain is analysed by an iterative technique using only HLQPN and LLQPN matrices. For details about the realisation of the algorithms we refer to [7, 9]. The approach allows to analyse Markov chains with several millions of states, see Sec. 4 for an example. The last step is the computation of the required performance measures from the stationary solution vector, exactly as for a flat analysis.

An approximate analysis technique, which is integrated in QPN-Tool, is based on repeated aggregation/disaggregation of LLQPNs. State space generation for this method is performed as described above. However, for an analysis the complete probability distribution vector is never generated. Instead, the LLQPNs and the HLQPN are analysed in isolation by assuming all other parts of the model are aggregated. Analysis of an isolated LLQPN/HLQPN yields new parameters for the aggregates, which are used for the isolated analysis of other parts. This process is iterated until a fixed point is reached. Although the technique is a heuristic it often yields sufficiently accurate results in a very short time, even for large models.

The implemented structured analysis module currently supports two levels of the hierarchy, hierarchical LLQPNs are simply flattened and subsequently analysed. The approach can be extended to use several hierarchical levels, but this presumably does not extend the size of solvable models, since most of the memory is used for the vector rather than for the generator matrix representation. Thus, from a solution point of view it is not necessary to support more than two levels, however, for specification convenience arbitrary hierarchies are highly recommended and are supported in QPN-Tool.

4 Example

The following example demonstrates the advantages of the structured analysis of HQPNs compared to conventional techniques.

The HLQPN of our example system is shown in Fig. 6: 5 workstations *Proc_1, ..., Proc_5* with a local disk each, and a special diskless workstation *Manager* with two processors and a joint processor sharing scheduling strategy. All workstations are connected via a bidirectional bus which has to be used exclusively.

The 5 workstations *Proc_1, ..., Proc_5* are identically specified by the subnet given in Fig. 7: A customer/token entering the subnet is forked into several jobs which are processed at the CPU. Several disk accesses might be necessary before a job completes its service at that workstation. A served job is modelled by a token of appropriate colour on the place *Done*. If all corresponding jobs have completed their service, they join again by firing transition *t_output* and the resulting customer/token may leave the subnet.

The system is used for an iteration problem, each workstation calculates a specific part of the iteration vector. This part is transmitted to the *Manager* via the *Bus*. When the *Manager* has received all parts of the new iteration vector, it updates its local copy and distributes the updated vector to all workstations via broadcasting. We assume that calculation of the fixed point takes a huge number of iterations and therefore only consider the iteration process, neglecting all convergence issues in our model.

The model is analysed for an increasing number of jobs running on the workstations. All experiments were performed on a workstation with 16MB main memory and 48MB virtual memory, time is measured in seconds as real time which is a more appropriate measure than CPU time. Table 1 and 2 summarise the results for some configurations. We assume that all workstations are identical, thus in the structured analysis approach we need to generate only matrices for workstations with different populations. It can be seen that the size of the state space grows rapidly with an increasing number of jobs in the system. The first column contains the total number of jobs, the second column in table 1 and 2 the number of tangible + vanishing states and the number of tangible states, respectively. The number of tangible states is identical for the conventional and structured approaches, however, in the structured approach vanishing states are eliminated locally, such that only the tangible part of the complete state space is generated. The number of tangible states equals the state space size of the Markov chain. For the conventional approach first all states, tangible and vanishing, have to be generated, subsequently the vanishing states are eliminated. For a structured analysis only HLQPN and LLQPN state spaces have to be generated and vanishing states can be eliminated locally. The largest state space which has to be handled for a structured analysis is the HLQPN state space consisting of 272 tangible and 682 vanishing states. The LLQPN state spaces are all smaller for the above parameters. The time needed to compute the state space and to eliminate vanishing markings is given in column four. Apart from the smallest configuration with only 5 jobs, the structured approach is much faster, in spite of the strictly sequential implementation of the structured state space generation, which means that the generation process is started several times which increases the effort for smaller state spaces. The generation time for the structured approach is nearly independent of the state space size, whereas the generation time for the conventional approach depends heavily on the number of states to be generated and the number of vanishing states to be eliminated.

# jobs	# tang. + vanish. states	non-zeros	generation time	time per iter.
5	3.56e+3	7.94e+3	7.60e+1	5.00e-1
10	4.59e+4	1.86e+5	2.90e+2	1.38e+1
12	1.05e+5	4.87e+5	5.97e+3	7.12e+2

Table 1. Results for conventional analysis

# jobs	# tang. states	non-zeros	generation time	time per iter.
5	1.32e+3	1.19e+3	1.69e+2	5.50e-1
10	2.29e+4	1.20e+3	1.79e+2	2.20e+0
12	5.60e+4	1.23e+3	2.17e+2	4.60e+0
15	2.10e+5	1.22e+3	1.82e+2	1.98e+1
20	1.30e+6	1.23e+3	1.85e+2	2.03e+2
23	3.30e+6	1.30e+3	2.55e+2	3.14e+3

Table 2. Results for structured analysis

Column three contains the number of non-zero elements which have to be stored to represent the generator matrix. For the structured approach the number of elements is nearly invariant for an increasing number of jobs. On a first glance at table 2 it might be surprising that we need more elements for 12 than for 15 jobs; however, this is caused by the different workstation populations for 12 jobs which requires the storage of two sets of matrices, one for workstations with 2 running jobs, and one for workstations with 3 running jobs. For 15 jobs all workstations receive 3 jobs. The storage of iteration and solution vectors is the limiting factor of structured analysis. The conventional approach requires a huge number of elements to be stored, although the generator matrix is sparse and we use, of course, a sparse storage scheme. The storage of the generator matrix becomes the limiting factor of the conventional approach. We were able to handle up to 12 jobs with the conventional approach, which equals a model with slightly more than 100,000 states; for larger populations the elimination of vanishing markings exhausts available memory. The structured approach allows to handle models with up to 23 jobs and more than three millions tangible states and additional vanishing states which are never generated in the complete state space. Of course, the solution of such a model is time consuming, if vectors do not fit into main memory.

The time needed for one iteration of the vector with the iteration matrix is given in column five for the conventional and structured approach, respectively. It is obvious that the structured approach takes a similar time for small state spaces and is much faster for larger models. The latter is caused by the extremely time consuming paging operations needed for the conventional method whenever the generator matrix does not fit into primary memory. The time required for a complete solution depends on the transition rates, the solution technique, the initial vector and the required accuracy. However, starting from an initial vector obtained by the approximation method as described above, the model can be analysed for a rather wide variety of parameters with between 40 and 300 iteration steps to yield an estimated accuracy of 10^{-6} for the solution vector. Thus it is possible to solve all models on a standard workstation, although the analysis of the largest configuration would take quite long. Using slightly more powerful machines would reduce the solution time to one night.

It should be noticed that the above model can be analysed more efficiently by exploiting symmetries in the model specification to generate first an exactly reduced Markov chain, which is afterwards solved using the structured approach. Examples for the reduction of a similar model are given in [3], the technique extends results on Stochastic Well-Formed Nets [10] and is described in [8]. Symmetry exploitation is currently not available in QPN-Tool, work towards an implementation is underway.

5 Conclusions

The new version of QPN-Tool supports the specification and efficient analysis of hierarchically combined Queueing Petri nets (HQPNs).
It attacks the two major problems of model-based system analysis:
Largeness of model specifications: HQPNs combine coloured GSPNs, queueing networks and offer hierarchical specification facilities. This yields specification advantages which result in a compact model specification: from coloured GSPNs we obtain simple description methods for synchronisation, fork and join-operations, from queueing networks we gain natural description methods for queues and scheduling strategies and finally the hierarchical approach results in the capability of reusing submodels and of keeping a clear and structured design. The specification of HQPNs is encouraged in QPN-Tool by a convenient graphical user interface. Even novice users get quickly acquainted with HQPNs.

A simple and convenient specification technique is very important for increasing user acceptance [22].

Largeness of state spaces: Quantitative analysis, especially numerical analysis of finite Markov chains, notoriously suffers from the so-called state space explosion problem. Modern tensor based iteration techniques extend the set of solvable state spaces by about one order of magnitude. This is possible if the model is structured adequately. QPN-Tool manages to exploit the hierarchical structure obtained by the hierarchical modelling specification for a structured, tensor based quantitative analysis.

A further advantage of HQPNs is that they combine the description of qualitative and quantitative system aspects in one modelling formalism. QPN-Tool supports these directions and again exploits the hierarchical structure of the model. It offers qualitative analysis algorithms based on Petri net theory for a so-called local analysis of isolated subnets and for a global analysis of a complete HQPN. This assists a user to identify well-defined parts/modules of his system and it especially facilitates error recognition.

Altogether QPN-Tool combines several modelling formalisms into a formalism, which is supported by a convenient graphical user interface, and combines qualitative and quantitative analysis techniques, which are able to exploit the hierarchical structure given by a model specification.

Future developments of the QPN-Tool are dedicated to the integration of simulative techniques. Further prospects concern exact and approximate aggregation techniques based on the hierarchical description and the exploitation of model symmetries. The integration of immediate queueing places [2] and queues with further scheduling strategies is planned as well [5]. At present the graphical user interface is being recoded for the X Window system.

References

1. M. Ajmone-Marsan, G. Balbo, and G. Conte. Performance models of multiprocessor systems. *MIT Press Series in Computer Science (1986).*
2. F. Bause. Queueing Petri Nets — a formalism for the combined qualitative and quantitative analysis of systems. In: *PNPM'93, IEEE Press (1993) 14-23.*
3. F. Bause, P. Buchholz, and P. Kemper. Hierarchically combined Queueing Petri Nets. In: *G. Cohen, J. P. Quadrat (eds.), 11th Int. Conference on Analysis and Optimizations of Systems, Springer LNCIS 199 (1994) 176-182.*
4. F. Bause and P. Kemper. QPN-Tool for the qualitative and quantitative analysis of Queueing Petri Nets. In: *G. Haring, G. Kotsis (eds.); Computer Performance Evaluation, Modelling Techniques and Tools 94, Springer LNCS 794 (1994).*
5. F. Bause, H. Kabutz, P. Kemper, P. Kritzinger. *SDL and Petri net performance analysis of Communicating systems.* 15th International Symposium on Protocol Specification, Testing and Verification, Warsaw (Poland), June 1995.
6. H. Beilner, J. Mäter, N. Weißenberg. Towards a performance modelling environment: news on HIT. In *R. Puigjaner (ed.), Proc. of the 4th Int. Conf. on Modelling Tools and Techniques for Comp. Perf. Eval., Plenum Publishers (1988).*
7. P. Buchholz. A hierarchical view of GCSPNs and its impact on qualitative and quantitative analysis. *J. of Parallel and Distributed Computing 15 (1992) 207-224.*

8. P. Buchholz. Aggregation and reduction techniques for hierarchical GCSPNs. In: *PNPM'93, IEEE Press (1993) 216-225.*
9. P. Buchholz. A class of hierarchical queueing networks and their analysis. *Queueing Systems 15 (1994) 59-80.*
10. G. Chiola, C. Dutheillet, G. Franceschini, S. Haddad. Stochastic well-formed coloured nets and multiprocessor modelling applications. *IEEE Trans. on Comp. 42 (1993).*
11. G. Chiola. GreatSPN 1.5 Software Architecture. In: *G. Balbo and G. Serazzi (eds.), Computer Performance Evaluation, North Holland (1992) 121-136.*
12. G. Ciardo, J. Muppala, K.S. Trivedi. SPNP: stochastic Petri net package. in *PNPM'89, IEEE Press (1989) 142-151.*
13. T. Demaria, G. Chiola, G. Bruno. Introducing a color formalism into Generalized Stochastic Petri nets. In: *Proc. of the 9th Int. Workshop on Application and Theory of Petri Nets (1988) 202-215.*
14. S. Donatelli. Superposed Generalized Stochastic Petri nets: definition and efficient solution. In *R. Valette (ed.), Application and Theory of Petri Nets 1994, Springer LNCS 815 (1994) 258-277.*
15. R. German, C. Kelling, A. Zimmermann, G. Hommel. TimeNET: a toolkit for evaluating non-markovian stochastic Petri nets. to appear in *Performance Evaluation (1995).*
16. P. Huber, K. Jensen, R.M. Shapiro. Hierarchies in coloured Petri nets. In: *G. Rozenberg (ed.), Adv. in Petri Nets 1990, Springer LNCS 483 (1991) 215-243.*
17. P. Kemper. Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net. In: *M. Ajmone-Marsan (ed.), Application and Theory of Petri Nets 93, Springer LNCS 691 (1993) 319-338.*
18. K. Lautenbach. Linear algebraic calculation of deadlocks and traps. In: *K. Voss, H.J. Genrich, and G. Rozenberg (eds.). Concurrency and Nets, Springer (1987).*
19. C. Lindemann. DSPNexpress: a software package for the efficient solution deterministic and stochastic Petri nets. to appear in *Performance Evaluation (1995).*
20. J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalized Petri net. *Application and Theory of Petri Nets, Selected Papers 1st, 2nd Europ. Workshop on Application and Theory of Petri Nets, 1981.*
21. B. Plateau, J.M. Forneau, K.H. Lee. PEPS: A package for solving complex Markov models of parallel systems. In *R. Puigjaner (ed.), Proc. of the 4th Int. Conf. on Modelling Tools and Techniques for Comp. Perf. Eval., Plenum Publishers (1988).*
22. D.A. Reed. Experimental analysis of parallel systems: techniques and open problems. In: *G. Haring, G. Kotsis (eds.); Computer Performance Evaluation, Modelling Techniques and Tools 94, Springer LNCS 794 (1994).*
23. W.H. Sanders, W.D. Obal, M.A. Qureshi, F.K. Widjanarko. The UltraSAN modelling environment. to appear in *Performance Evaluation (1995).*
24. W.J. Stewart. Introduction to the numerical solution of Markov chains. *Princeton University Press (1994).*

This article was processed using the LaTeX macro package with LLNCS style