QPN-Tool for Qualitative and Quantitative Analysis of Queueing Petri Nets

Falko Bause, Peter Kemper

Lehrstuhl Informatik IV
Universit¨at Dortmund
44221 Dortmund
Germany

Abstract. Synchronisation and concurrency aspects as well as sharing of resources are common features of distributed systems. Modelling the last aspect, especially the scheduling strategy amongst competing jobs, can be extremely hard using (Coloured) Generalized Stochastic Petri nets (CGSPNs).
Queueing Petri nets (QPNs) provide additional elements for a convenient specification of such queueing situations. QPNs can be used for qualitative analysis employing efficient techniques from Petri net theory, and performance analysis (quantitative analysis) exploiting Markovian analysis algorithms.
QPN-Tool supports both forms of analysis and offers a convenient graphical interface enabling also unexperienced users to specify and analyse their system using the QPN model world.

1 Introduction

System analysis is often done with respect to qualitative and quantitative aspects. E.g. one feature of a fault-tolerant computer is that it will eventually recover from an error which is a qualitative property of the system. A designer of such a system is surely also interested in the time needed for recovery, a quantitative property.

Several formalisms have been developed for modelling and analysing qualitative properties. Petri nets (PNs) [17, 22] belong to these formalism. They have been proved to be suitable for representation of concurrency and synchronisation aspects in modern distributed systems. Since PNs do not involve any notion of time, temporal descriptions have been incorporated to render them suitable also for quantitative analysis leading to Timed and Stochastic Petri nets. A well-known representative of this class are Generalized Stochastic Petri nets (GSPNs) [1, 2], which have been used for modelling a variety of systems [18–21].

Apart from concurrency and synchronisation another characteristic of distributed systems is sharing of common resources. Modelling this aspect, especially the scheduling rule, of a system using (GS)PN elements is quite difficult and leads to large and complex models [3].

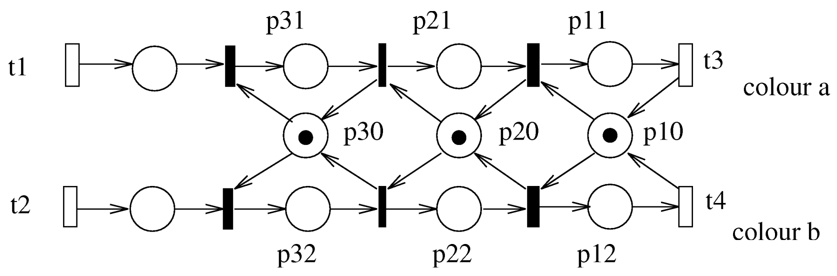


Fig. 1. GSPN model of a FCFS queue

Consider, e.g., a queue where 2 colours of tokens arrive according to exponentially distributed interarrival times and whose service times are exponentially distributed. If the scheduling strategy is FCFS, one has to encode the colour of the token in each position of the queue. Assume that an upper bound for the number of tokens in the queue is given, e.g. 3, then the GSPN in Fig. 1 would model the queue accurately. Transitions t_1 and t_2 model the arrival of a token of either colour and transitions t_3 and t_4 model the service of the token in front of the queue. The places p_{i1} and p_{i2} represent position i of the queue and the place p_{i0} ensures that this position is occupied by at most one token. Since entering a position is modelled by immediate transitions, a token entering position 3 of the queue will immediately advance to position 2 and 1, if they are free. This way of modelling a FCFS queue with GSPN elements works fine if an upper bound for the number of tokens is known *a priori*. Performing several experiments with different initial markings will necessitate a modification of the GSPN model of the queue for each experiment. If there is no upper bound known beforehand it is even more difficult. Other service times and scheduling strategies lead to very complex models. If the service time of a queue is specified by, e.g. a Coxian distribution and the scheduling strategy is Last Come First Served - Preemptive Resume, it becomes just about impossible to model such a queue with a GSPN.

A popular modelling world to represent resource sharing are queues [7, 15], where system behaviour can be modelled in a compact way. Receiving the benefits of both modelling worlds GSPNs have been enhanced by the usual descriptions of queues leading to a new model, Queueing Petri nets (QPNs) [3]. Queues can be directly integrated into Coloured GSPNs (CGSPNs) by associating them with the places of the net, since a basic property of queues is that customers entering the queue will eventually leave it. QPNs offer a specification paradigm where concurrency and synchronisation aspects are described by (CGS)PN elements and resource sharing is modelled by queues giving a structured model of a system.

Since analysis of modern systems can not be done without proper tool support, we have developed a program package (QPN-Tool) offering a convenient graphical user interface and several analysis algorithms for QPN models. Instead of specifying the transitions of a Coloured GSPN by predicates or functions, QPN-Tool automatically provides a local unfolding so that also unexperienced users can specify their QPN models. Furthermore QPN-Tool offers a variety of PN algorithms and automatically determines quantitative properties of the system employing efficient algorithms from PN and Markov theory. In Sect. 2 we introduce QPNs and the QPN-Tool is described in Sect. 3. A short comparison with other tools is given in Sect. 4.

2 The QPN world

Queueing Petri Nets (QPNs) [3,4,6] combine Coloured Generalized Stochastic Petri Nets (CGSPNs) [10] with Queueing Networks (QNs) by hiding stations in special places of the CGSPN which are called *timed places.* The structure of a QPN is determined by a CGSPN with two types of places and transitions:

ordinary place An ordinary place is equivalent to a place in a Coloured Petri net. Tokens fired onto such a place are immediately available for the corresponding output transitions.

timed place A timed place contains a queue and a depository. A token which is fired on a timed place is inserted into the queue according to a scheduling strategy. The scheduling strategy determines which tokens in the queue are served. Each colour has an individual service time distribution of Coxian type. After receiving service the token moves to a depository, where it is available to the place's output transitions. Figure 2 presents a timed place and its graphical shorthand notation. A timed place can be regarded as a short notation of a complex CGSPN subnet, which models a Queueing Network service station. Scheduling strategies like FCFS, which concern the order of arrival, require a ranking of token colours to handle bulk arrivals. In case of a bulk arrival all tokens are separately inserted in succession into the queue in zero time. A token of the colour with the highest rank is inserted first.

immediate transition An enabled immediate transition fires according to one of its colours without any delay in zero time. Any of its colours has a so called 'firing frequency', which allows to compute firing probabilities in case of concurrently enabled immediate transitions.

timed transition An enabled timed transition fires after a certain delay. This delay is determined by a colour-specific exponential distribution. Firing of timed transitions has a lower priority than firing of immediate transitions. Hence no timed transition can fire if an immediate transition is enabled. Like in GSPNs timed transitions obey a 'race policy' to solve conflicts. Firing of a transition is always an atomic action.

The following example focuses on the illustration of the different elements a QPN can contain and not on modelling anything of practical relevance.

Example 1. Our model describes a situation such that two types of jobs - they are classified as *light* and *heavy* - require service at a first service station, then

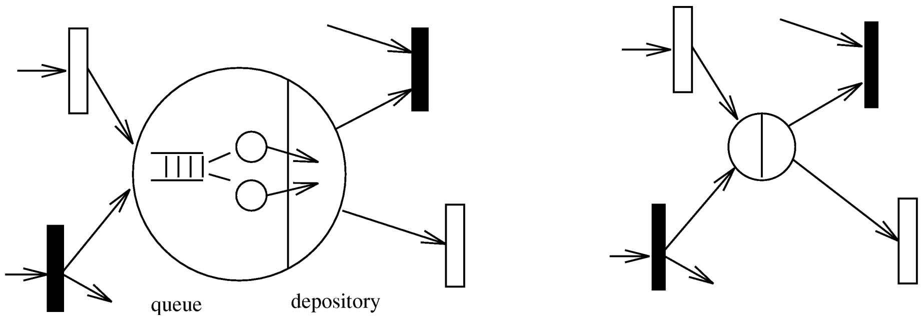
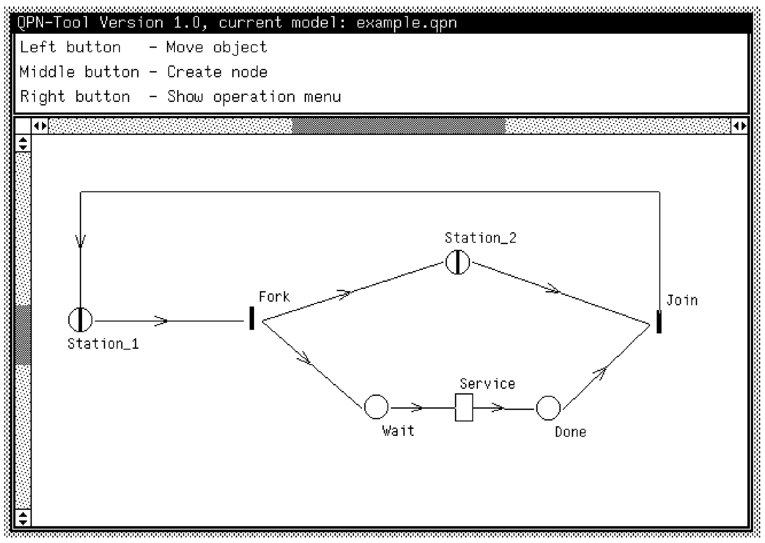


Fig. 2. Timed place in a QPN and its shorthand notation


QPN-Tool Version 1.0, current model: example.qpn

```
Left button   - Move object
Middle button - Create node
Right button  - Show operation menu
```

Fig. 3. Example of a QPN

fork into two subjobs which require service from different resources before they join and then start at the first station again. Thus these jobs are never done. Figure 3 shows the corresponding QPN with timed places *Station_1* and *Station_2*, ordinary places *Wait* and *Done*, immediate transitions *Fork* and *Join* and a timed transition *Service*. It is a net with two colours at each place and transition. The scheduling strategy of *Station_1* is processor sharing (PS) and *Station_2* serves jobs according to their arrival (FCFS). For an initial marking 3 tokens of colour '*light*' and 4 tokens of colour '*heavy*' are supposed at *Station_1*, all other places are empty. Service time distribution in *Station_1* is Coxian with 2 phases for colour '*heavy*', exponential for colour '*light*' which equals a Coxian distribution with 1 phase. Actual values for rates and probabilities shall not be of further interest here.

Every QPN describes a stochastic process. A state of this process is determined by the cartesian product of the state descriptions at all timed places and the number of tokens at ordinary places with respect to their colours. If the service time within a timed place is modelled by an appropriate distribution, e.g. Cox-distribution, Markov-chain based analysis of the QPN is possible.

The state space is partitioned into two types of states similar to GSPNs (cf. [1]):

vanishing states Firing of an immediate transition has a higher priority than any other change of state. Thus the stochastic process immediately leaves a state in which an immediate transition is enabled. If several immediate transitions are enabled, the one which fires first is determined by the firing probability. Firing probabilities are deduced from firing frequencies by relating a firing frequency to the sum of firing frequencies of all enabled transitions.

tangible states If no immediate transition is enabled, firing of a timed transition or serving a token within a timed place can cause a change of state. The time for this change is determined by an exponential distribution in case of firing a timed transition or by the corresponding (exponential stage of the) service time distribution.

The initial marking of the QPN gives the initial state of its stochastic process under the assumption that initially all tokens on timed places are situated on the corresponding depository.

3 QPN-Tool

Specification and analysis of QPNs require appropriate tool support. This section contains a description of QPN-Tool, which is developed at LS Informatik IV, University of Dortmund. Furthermore a brief introduction into the qualitative and quantitative analysis of QPNs is given.

QPN-Tool is a prototype which is implemented in C and is executable on Sun3, Sun4-machines with Sunview or OpenWindows. It contains a graphical user interface and a variety of analysis algorithms for qualitative and quantitative analysis of QPNs. Figure 4 describes the modular structure of QPN-Tool. A brief description of its different modules follows.

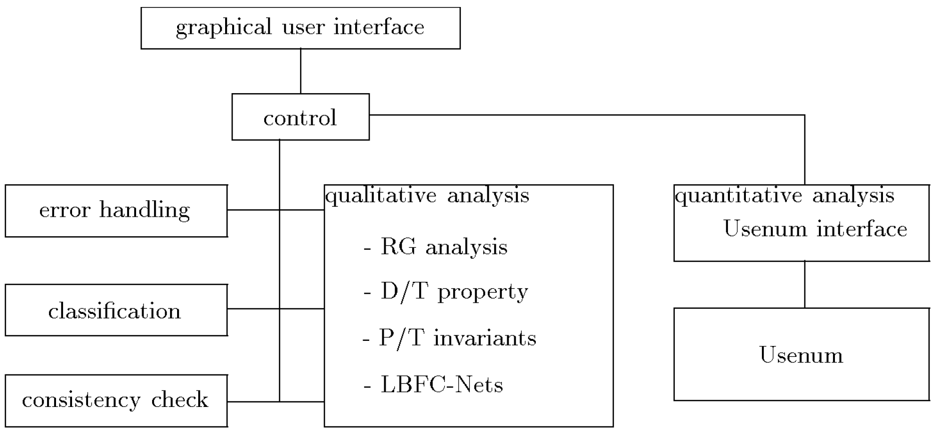


Fig. 4. Modular structure of QPN-Tool

3.1 Graphical user interface

The graphical user interface manages the complete user interaction. It supports:

- specification of QPN models,
- specification of analysis tasks, selection of analysis algorithms and required performance measures,
- presentation of results concerning classification of QPNs and results of qualitative and quantitative analysis
- aggregation of detailed information on probably undesired net properties, e.g. brief description of a firing sequence which leads to a deadlock.

The model description process consists of several main steps.

1. The net's graphical representation is clearly dominated by the Coloured Petri net part. This part is specified by creating and positioning places and transitions and establishing their connections by directed arcs. Figure 3 shows the net corresponding to Example 1.

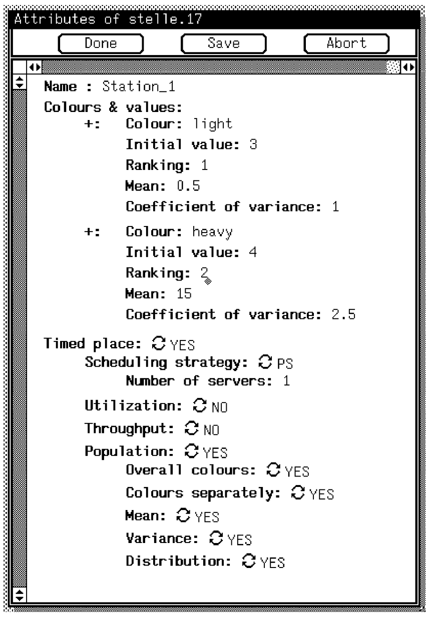
2. Attributes of places and transitions have to be set. This includes entering the different colours in a list, choosing the type of transitions and places (timed or immediate, resp. timed or ordinary) and fixing the number of tokens for the initial marking. Especially timed places require some additional information:

1. for each colour
   (a) its rank according to bulk arrivals
   (b) its service time distribution specified by its mean and coefficient of variance. This service time distribution is approximated by a Cox distribution as described in [8].
2. scheduling strategy
   Presently available scheduling strategies are FCFS, LCFS-Pr, PS and Infinite Server (IS).
3. number of servers
4. performance figures
   to be determined in quantitative analysis (cf. Sec. 3.2.4).

Figure 5 displays attributes for *Station_1* of example 1.

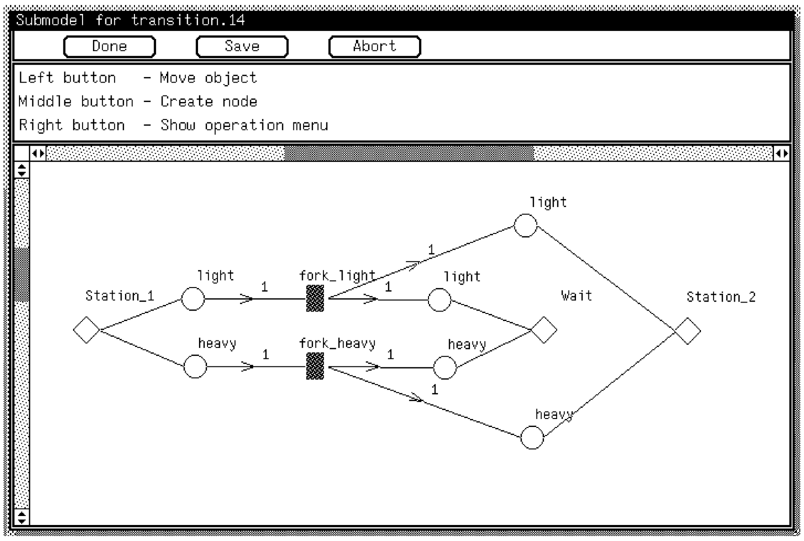
3. The incidence functions of the Coloured Petri net (cf. [10]) have to be specified. This is possible in a graphical submodel at each transition. Such a submodel describes the locally unfolded net regarding a single transition and its input and output places. A special feature of QPN-Tool fills the 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.

Figure 6 shows the submodel of transition *Fork* in Fig. 3. A rhomb represents a coloured place. Colours of a place are represented by circles which are connected to its corresponding place by lines. Bars display colours of the transition whose submodel is regarded. Obviously transition colour '*fork_light*' takes one '*light*'



```
Name : Station_1
Colours & values:
     +:   Colour: light
          Initial value: 3
          Ranking: 1
          Mean: 0.5
          Coefficient of variance: 1
     +:   Colour: heavy
          Initial value: 4
          Ranking: 2
          Mean: 15
          Coefficient of variance: 2.5
Timed place: YES
     Scheduling strategy: PS
          Number of servers: 1
     Utilization: NO
     Throughput: NO
     Population: YES
          Overall colours: YES
          Colours separately: YES
          Mean: YES
          Variance: YES
          Distribution: YES
```

Fig. 5. Attributes of timed place *Station_1*



```
Left button   - Move object
Middle button - Create node
Right button  - Show operation menu
```

Fig. 6. Locally unfolded net at transition *Fork*

token from place '*Station_1*' and puts on places '*Wait*' and '*Station_2*' one token on each of the corresponding colour '*light*'. Transition colour '*fork_heavy*' does the same for '*heavy*' tokens.
Locally unfolding a net has certain advantages:

- It allows a detailed view on single active components with their corresponding environments.
- The Petri-net-type formalism in locally unfoldings and the whole net is homogeneous.
- Automatic generation of graphical objects in a local unfolding supports a convenient specification.
- Furthermore generating available colours of input/output places as well as transition colours automatically avoids inconsistent specifications, because the set of colours which can be used and should be used are presented completely. Default positions reflect the common reading direction: input places and their colours to the left, transition colours in the middle, output places and their colours to the right.
- If several transitions have identical submodels, it is sufficient to specify just one. Sharing a submodel is possible as well as copying it.

Since locally unfolding is used within QPN-Tool, it combines a comfortable description technique with a clear presentation.

3.2 Analysis techniques

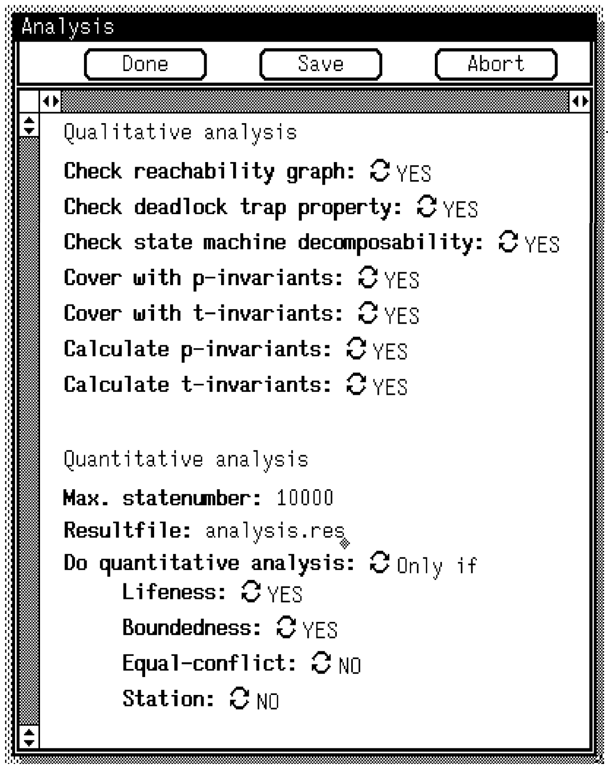
Typically analysis goals either refer to qualitative properties, e.g. absence of deadlocks, liveness or boundedness, or determination of performance measures. Within QPN-Tool qualitative properties are investigated by a so-called qualitative analysis which is based on Petri net theory. Performance measures are computed by analysis of the corresponding Markov-chain, which is called quantitative analysis. These analysis techniques are briefly described in the following.

Before qualitative analysis a consistency check is performed and the QPN is classified as described below.

3.2.1 Consistency check This module checks a QPN for specification inconsistencies. These could be naming inconsistencies between colour names of a place or transition and its corresponding colours in a locally unfolded transition. This type of error can be avoided by automatic generation of net components in locally unfolded transitions.

Another common error is: although the QPN appears as a connected graph, it is possible that for certain colours a place or transition happens to be a source or sink. This affects boundedness or liveness of a QPN. Thus it is checked and a user information is produced.

3.2.2 Classification Classification aims at the embedded CPN. This net is unfolded to an uncoloured Place/Transition net and classified in terms of: marked graph, state machine, free choice, extended free choice, simple or extended simple. This classification supports the choice of a suitable analysis algorithm for qualitative analysis, because for certain net classes special algorithms are available.



```
Qualitative analysis
Check reachability graph: YES
Check deadlock trap property: YES
Check state machine decomposability: YES
Cover with p-invariants: YES
Cover with t-invariants: YES
Calculate p-invariants: YES
Calculate t-invariants: YES

Quantitative analysis
Max. statenumber: 10000
Resultfile: analysis.res
Do quantitative analysis: Only if
     Lifeness: YES
     Boundedness: YES
     Equal-conflict: NO
     Station: NO
```

Fig. 7. Selection of analysis algorithms

3.2.3 Qualitative analysis Figure 7 presents the selection of qualitative analysis algorithms implemented in QPN-Tool. Apart from 'classical' algorithms like reachability graph analysis and calculation of P- and T-invariants, rather new algorithms for special net classes are offered. Qualitative analysis within QPN-Tool aims at liveness and boundedness. If the QPN does not have these properties, information of the employed algorithm is extracted in order to demonstrate the reason for a QPN being unbounded or not live. All implemented qualitative analysis algorithms are based on Petri net theory and ignore timing aspects as firing delays and frequencies and interpret timed places as ordinary

places. Properties of this ‘untimed’ QPN carry over to the timed QPN under certain circumstances, if conditions Equal-conflict and Station are satisfied (cf. [3]). Condition Equal-conflict demands that only transitions of the same kind, either timed or immediate, are in conflict and condition Station states that the scheduling strategy has to be of a type like PS or IS. If the QPN is live, bounded and unveils an extended free choice net-structure these conditions are sufficient for the existence of the steady-state distribution.
The choice of algorithms contain:

reachability graph analysis The algorithm generates the reachability graph and recognises firing sequences of transitions which lead to unbounded markings. The reachability graph is checked and it is determined whether the net is bounded and live. If this is not the case a firing sequence is presented which demonstrates unboundedness or non-liveness.

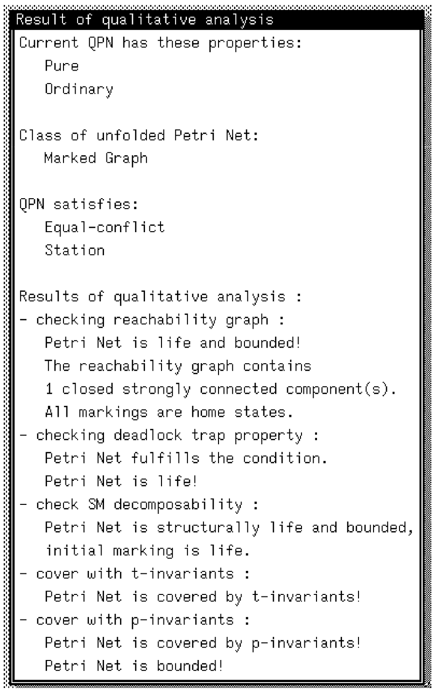
computation of P- or T-invariants A system of 'base vectors' for all positive P- and T-invariants is computed.

cover of P- or T-invariants The algorithm checks whether the net can be covered by positive P- or T-invariant. If not, the set of uncovered places, resp. transitions is presented.

deadlock/trap condition For the class of simple nets it is possible to ensure liveness by checking the deadlock/trap condition. The algorithm is taken from [16] and generates the set of minimal deadlocks and checks if any minimal deadlock contains a marked trap.

check state-machine-decomposability The algorithm is taken from [13, 14]. It allows to recognise live and bounded Free-Choice nets by checking the net structure. This is highly efficient compared to other analysis algorithms.

The selection of algorithms allows to exploit the advantages of an algorithm for the particular case.



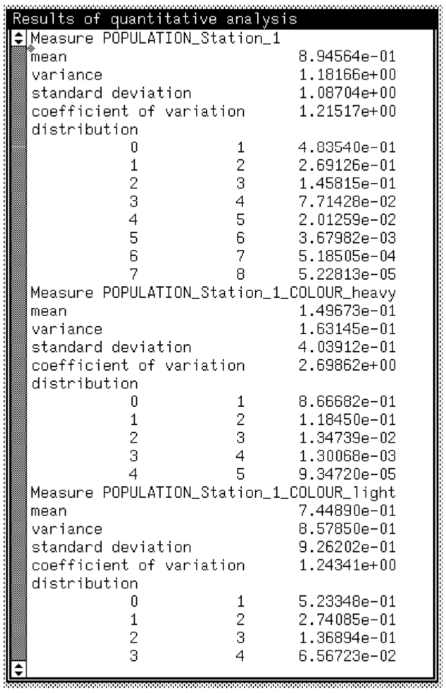
```
Result of qualitative analysis
Current QPN has these properties:
   Pure
   Ordinary

Class of unfolded Petri Net:
   Marked Graph

QPN satisfies:
   Equal-conflict
   Station

Results of qualitative analysis :
- checking reachability graph :
   Petri Net is life and bounded!
   The reachability graph contains
   1 closed strongly connected component(s).
   All markings are home states.
- checking deadlock trap property :
   Petri Net fulfills the condition.
   Petri Net is life!
- check SM decomposability :
   Petri Net is structurally life and bounded,
   initial marking is life.
- cover with t-invariants :
   Petri Net is covered by t-invariants!
- cover with p-invariants :
   Petri Net is covered by p-invariants!
   Petri Net is bounded!
```

Fig. 8. Result window of classification and qualitative analysis



```
Results of quantitative analysis
Measure POPULATION_Station_1
mean                              8.94564e-01
variance                          1.18166e+00
standard deviation                1.08704e+00
coefficient of variation          1.21517e+00
distribution
             0            1       4.83540e-01
             1            2       2.69126e-01
             2            3       1.45815e-01
             3            4       7.71428e-02
             4            5       2.01259e-02
             5            6       3.67982e-03
             6            7       5.18505e-04
             7            8       5.22813e-05
Measure POPULATION_Station_1_COLOUR_heavy
mean                              1.49673e-01
variance                          1.63145e-01
standard deviation                4.03912e-01
coefficient of variation          2.69862e+00
distribution
             0            1       8.66682e-01
             1            2       1.18450e-01
             2            3       1.34739e-02
             3            4       1.30068e-03
             4            5       9.34720e-05
Measure POPULATION_Station_1_COLOUR_light
mean                              7.44890e-01
variance                          8.57850e-01
standard deviation                9.26202e-01
coefficient of variation          1.24341e+00
distribution
             0            1       5.23348e-01
             1            2       2.74085e-01
             2            3       1.36894e-01
             3            4       6.56723e-02
```

Fig. 9. Result window of quantitative analysis

3.2.4 Quantitative analysis

Quantitative Analysis is pursued with the objective of assessing performance properties for a QPN. Different performance measures are offered for

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

The calculation of performance measures can result in mean value, variance and distribution. They can be computed for all colours of a place separately or aggregated overall colours. They are easily specified by setting appropriate attributes of the corresponding place, see Fig. 5.

The employed analysis technique maps the specified QPN 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. For calculating a state descriptor for FCFS stations an upper bound of the queue length has to be determined. This is performed automatically during qualitative analysis by checking the reachability graph or an appropriate P-invariant.

QPN-Tool performs quantitative analysis based on Usenum. This is a tool developed at LS Informatik IV which is specially designed for numerical analysis of finite Markov chains, see [9]. Usenum's duties include three main steps:

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

It is able to handle state spaces with more than $100,000$ states and offers different algorithms for the calculation of the steady state distribution, e.g. Grassmann's algorithm, JOR, SOR.

4 Comparison

For recognising limitations of QPN-Tool a quick glance at similar and well-known existing tools might be helpful: GreatSPN [11] from the Universita di Torino, Italy, and SPNP [12] from Duke University, Durham, USA.

GreatSPN supports the specification and analysis of GSPNs and DSPNs. It provides a graphical interface which allows the modelling of only uncoloured nets but including inhibitor arcs, marking dependent rates and probabilities. Difficulties in describing non-trivial queueing situations and scheduling strategies as mentioned in Sect. 1 occur. GreatSPN offers an ample variety of algorithms for qualitative analysis which contains reachability graph analysis, computation of invariants, deadlocks, and traps, and the (inverse) token game. Net properties are nicely animated on its graphical representation. For quantitative analysis, Markov-chain based transient and steady-state analysis is provided as well as simulation. Output measures have to be textually defined. Compared to GreatSPN, future versions of QPN-Tool should be able to handle inhibitor arcs, marking dependent rates and probabilities, and deterministic times. Analysis algorithms of QPN-Tool do not contain the token game, transient analysis and simulation yet.

SPNP is based on the analysis of Markov reward models. Its textual interface is closely related to the programming language 'C', the language SPNP is implemented in. It allows marking dependent arcs, marking dependent enabling functions and general priorities. According to its descriptive power only reachability-graph based qualitative analysis is performed. Its main focus is on quantitative analysis based on Markov reward models (transient and steady state analysis). Output measures are specified by user-defined C-functions supported by a set of predefined functions. An automated sensitivity analysis is offered which derives different CTMCs from a fixed state space by variation of an independent parameter μ for firing rates and probabilities. Compared to QPN-Tool the modelling process in SPNP tends to be a programming process with a strict focus on Markov reward process analysis and few qualitative analysis features as a debugging aid. A variety of qualitative analysis algorithms like in QPN-Tool or GreatSPN is not given in SPNP. As far as quantitative analysis is concerned, SPNP differs from QPN-Tool by its transient analysis, automated sensitivity analysis and its ability to handle general reward specifications.

This comparison is not supposed to be exhaustive or to give a complete characterisation of GreatSPN and SPNP. We just wanted to demonstrate limits of QPN-Tool to draw the following conclusions.

5 Conclusions

QPNs are suitable for modelling synchronisation and concurrency situations as well as sharing of resources which appear in most distributed systems. The great benefit is that a user is not forced to model queues by ordinary (CGS)PN elements thus simplifying the description of systems. Timed places can be viewed as simple parametrisable subnets of a hierarchically specified model.

QPN-Tool offers a convenient graphical user interface. The automatic local unfolding of transitions enables also unexperienced users to get quickly acquainted with the QPN model world. The tool also offers many algorithms for efficient qualitative and quantitative analysis of QPN models.

Future developments are directed to extend this set of analysis algorithms towards transient analysis, simulation and furthermore to integrate hierarchical description and analysis techniques as proposed in [5].

References

1. M. Ajmone-Marsan, G. Balbo, G. Conti. *Performance Models of Multiprocessor Systems.* MIT Press Series in Computer Science, 1986.
2. M. Ajmone-Marsan, G. Conti, G. Balbo. A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

3. F. Bause. Queueing Petri Nets: a formalism for the combined qualitative and quantitative analysis of systems. In [21].
4. F. Bause, H. Beilner. Eine Modellwelt zur Integration von Warteschlangen- und Petri-Netz-Modellen. In *Proceedings of the 5th GI/ITG-Fachtagung, Messung, Modellierung und Bewertung von Rechensystemen und Netzen*, pages 190–204. Gesellschaft für Informatik (GI), Braunschweig (Germany), September 1989.
5. F. Bause, P. Buchholz, P. Kemper. Hierarchically Combined Queueing Petri Nets. 11th International Conference on Analysis and Optimizations of Systems, Discrete Event Systems, Sophia-Antipolis (France), June 1994.
6. F. Bause, P. Kemper. Queueing Petri nets. In *Proceedings of the 3rd Fachtagung Entwurf komplexer Automatisierungssysteme, Braunschweig.* Technische Universität Braunschweig (Germany), May 1993.
7. E. Gelenbe, G. Pujolle. *Introduction to Queueing Networks.* John Wiley & Sons, 1987.
8. P. Buchholz. Die strukturierte Analyse Markovscher Modelle. Informatik-Fachberichte, 282, Springer, 1991.
9. P. Buchholz, J. Dunkel, B. Müller-Clostermann, M. Sczittnick, S. Zäske. *Quantitative Systemanalyse mit Markovschen Ketten. Eine Darstellung für Informatiker und Ingenieure* Teubner-Verlag, to be published.
10. G. Chiola, G. Bruno, T. Demaria. Introducing a Color Formalism into Generalized Stochastic Petri Nets. In *Proceedings of the 9th International Workshop on Application and Theory of Petri Nets*, Venice (Italy), pp 202-215, 1988.
11. G. Chiola. GreatSPN 1.5 Software Architecture. In *Proceedings of the 5th International Conference Modeling Techniques and Tools for Computer Performance Evaluation*, Torino (Italy), Feb. 1991.
12. G. Ciardo, J. Muppala, K. Trivedi. SPNP: Stochastic Petri Net Package. In [19].
13. P. Kemper. Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net. In M. Ajmone-Marsan, editor, *Application and Theory of Petri Nets 1993*, LNCS 691, pages 319–338, Berlin, 1993. Springer.
14. P. Kemper, F. Bause. An efficient polynomial-time algorithm to decide liveness and boundedness of free-choice nets. In K. Jensen, editor, *Application and Theory of Petri Nets 1992*, LNCS 616, pages 263–278, Berlin, 1992. Springer.
15. L. Kleinrock. *Queueing Systems. Volume 1: Theory.* John Wiley and Sons, 1975.
16. K. Lautenbach. Linear algebraic calculation of deadlocks and traps. In K. Voss, H.J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets, Advances of Petri Nets*, Berlin, 1987. Springer.
17. J.L. Peterson. *Petri Nets and the Modelling of Systems.* MIT Press Series in Computer Science, 1981.
18. *Proceedings of the 2nd International Workshop on Petri Nets and Performance Models, Madison (USA).* IEEE Computer Society Press, 1987.
19. *Proceedings of the 3rd International Workshop on Petri Nets and Performance Models, Kyoto (Japan).* IEEE Computer Society Press, 1989.
20. *Proceedings of the 4th International Workshop on Petri Nets and Performance Models, Melbourne (Australia).* IEEE Computer Society Press, 1991.
21. *Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse (France).* IEEE Computer Society Press, 1993.
22. W. Reisig. *Petri Nets. An Introduction*, volume 4. EATCS Monographs on Theoretical Computer Science, Berlin, 1985, Springer.

A C K N O W L E D G E M E N T S

The authors are grateful for some valuable comments made by several anonymous referees.