APPROXIMATION METHODS FOR STOCHASTIC PETRI NETS

NAGW-1333

by

Hauke Jörg Jungnitz

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

May 1992

CIRSSE REPORT #119

CONTENTS

ACKNOWLEDGMENT

I am thankful to my advisor, Alan Desrochers, for his guidance and help. During my years at RPI, I had the chance to develop and work on my own ideas. I would also like to thank my committee members for their suggestions and help during the preparation of the thesis.

Of the many people at RPI who contributed directly or indirectly, I can only mention a few: Alessandro Giua patiently read through the entire thesis, pointed out many inconsistencies and helped with many discussions, Don Lefebvre was of great help with measurements of the case study. I also want to mention my office mates who were great companions along the way.

I am deeply indebted to Manuel Silva, who invited me to the University of Zaragoza, Spain. Every discussion with him produced new ideas and encouragement. Without my stay in Spain, this thesis would not exist in its present form. In Zaragoza, I would also like to thank José Manuel Colom for our many discussion and Beatriz Sánchez for our joint work.

I am thankful to the Center for Intelligent Robotics System for Space Exploration (CIRSSE), who offered me a graduate assistantship under NASA Grant # NAGW-1333 and IBM who also provided financial support. I would also like to acknowledge the Siemens International Scholarship, which helped to finance some of my travel.

I am thankful to Kishor Trivedi's for providing the SPNP program and G. Chiola's for providing the GreatSPN software package.

Finally, I would like to thank my parents for the support received throughout.

ABSTRACT

Stochastic Marked Graphs are a concurrent decision free formalism provided with a powerful synchronization mechanism generalizing conventional Fork Join Queueing Networks. In some particular cases the analysis of the throughput can be done analytically. Otherwise the analysis suffers from the classical state explosion problem.

Embedded in the divide and conquer paradigm, this thesis introduces approximation techniques for the analysis of stochastic marked graphs and Macroplace/-Macrotransition-nets (MPMT-nets), a new subclass which is introduced in this thesis. MPMT-nets are a subclass of Petri nets that allow limited choice, concurrency and sharing of resources. The modeling power of MPMT is much larger than that of marked graphs, e.g., MPMT-nets can model manufacturing flow lines with unreliable machines and dataflow graphs where choice and synchronization occur.

The basic idea leads to the notion of a cut to split the original net system into two subnets. The cuts lead to two aggregated net systems where one of the subnets is reduced to a single transition. A further reduction leads to a basic skeleton. The generalization of the idea leads to multiple cuts, where single cuts can be applied recursively leading to a hierarchical decomposition.

Based on the decomposition, a response time approximation technique for the performance analysis is introduced. Also, delay equivalence, which has previously been introduced in the context of marked graphs by Woodside et al., Marie's method and flow equivalent aggregation are applied to the aggregated net systems.

The experimental results show that response time approximation converges quickly and shows reasonable accuracy in most cases. The convergence of Marie's is slower, but the accuracy is generally better. Delay equivalence often fails to converge, while flow equivalent aggregation can lead to potentially bad results if a strong dependence of the mean completion time on the interarrival process exists.

CHAPTER 1
Introduction

1.1 Motivation

Manufacturing systems, computer systems and robotics systems cannot be described by differential or difference equations. System changes happen in random intervals of time, and are therefore stochastic in their nature. These systems are commonly referred to as discrete event dynamic systems (DEDS). Modeling, control and the analysis of DEDS represents a big challenge. Robotics systems composed of multiple arms, a vision system, force/torque sensor and control computer for example, are discrete event in their operation and have a rather large number of interacting components. It is the interaction of the subsystems coupled with the discrete nature of their dynamics which makes the modeling and performance analysis challenging [AJD90a].

In the modeling and analysis of DEDS there is a trade-off between the ability of how realistically our model is able to represent the actual system (modeling power), the cost and effort involved in the modeling process (economical considerations) and the *analytical tractability*, i.e., the ease of analysis of our model. Ideally we would like to have a single model, which can express the following characteristics in one representation.

- **Logical properties**

  1. *Asynchronous operations.* The model should be concerned with the independent "internal clocks".
  2. *Concurrency and parallelism.* Several activities can take place at the same time.

3. *Conflict.* This may occur when two or more processes require the same resource or a part can take several paths through the system.
4. *Synchronic distance.* How often can transition A fire before transition B has to fire.
5. *Deadlock.* A state can be reached where none of the processes can continue. For example this can happen with the sharing of a resource between two processes, e.g., two robots want to acquire a vision system simultaneously.
6. *Event driven.* Since operations can occur concurrently, the order of events is not necessarily unique.

- **Temporal properties** such as the duration of activities and timeout.
- **Operational properties** such as real time scheduling using mathematical or heuristic models and high level decision making.

Figure 1.1 shows the hierarchy of the modeling process. As we move to the top of the pyramid, we can increasingly model the desired characteristics of the system.

At the bottom of the modeling hierarchy are product form queuing networks (PFQNs) which are computationally very easy to solve, but this computational saving has a drawback: classical PFQNs (Jackson QNs, BCMP QNs) cannot model blocking, finite buffer size, synchronization and control over the dispatching of customers. Therefore, their modeling power is low. Because of their computational ease, PFQNs are most useful for the initial rough design and analysis of a system.

Petri nets (PNs) allow the modeling of the logical properties of the system. Depending on the net subclass, polynomial time algorithms exist for the analysis of liveness, boundedness and deadlock. The incorporation of time leads to generalized stochastic Petri nets (GSPNs) where transitions can fire in zero time (immediate transitions) or after an exponentially distributed amount of time (timed transitions).

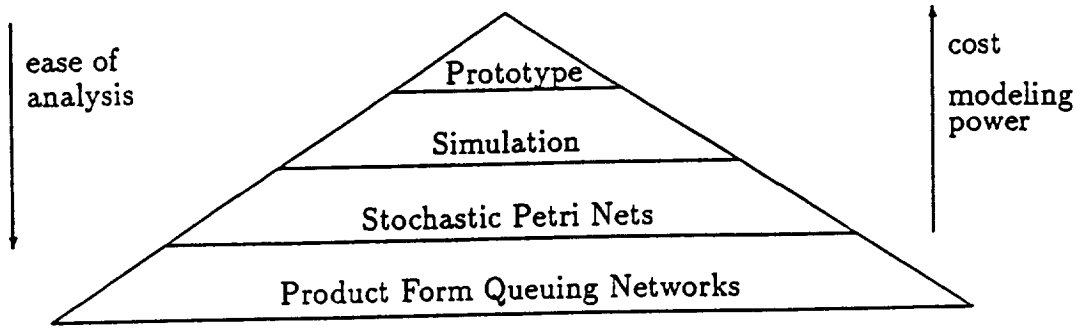


Figure 1.1: Hierarchical Representation of the Modeling Process

The underlying assumption is that the system is memoryless (Markovian), i.e., the next state depends *only* on the current state and *not* on the history of the previous states which were visited, or how long the system has been in the current state. The only continuous probability distribution which satisfies the Markovian property is the exponential distribution.

GSPNs can be interpreted as a generalization of traditional queuing networks (QNs) allowing synchronization, fork, split, choice and timeout constructs (see [VZL87] for a comparison of the modeling power between QNs and GSPNs). In order to determine the performance measures, the underlying continuous time Markov chain (CTMC) is constructed from the reachability graph and the corresponding system of linear equations is solved.

GSPNs can model a system with considerable level of detail. The major drawback for the use of GSPNs for the performance analysis is the problem of *state space explosion*. Even a moderately sized system can have thousands of states, and the solution of the CTMC becomes increasingly space and time consuming. Using more powerful computers or more efficient algorithms would help, but state space explosion is *exponential*, while faster algorithms [JGD91] or increased memory are a polynomial improvement at best. Ultimately, it is the problem of state explosion which renders the analysis of complex systems using GSPNs difficult. GSPNs

have been used for the analysis of computer systems [Mol82, Beo85], flexible manufacturing systems [AJ88, KDD88, BCFR87, CCS90] and manufacturing message specifications [WJG91]. GSPNs are most useful in the intermediate design state where various options are being explored, e.g., different buffer sizes, machining or processor speeds can easily be explored. Once a PN model has been found, it can also be used as the underlying structure for the simulation model [HS89, BC90].

Further up in the hierarchy is simulation, one of the most widely used techniques in operations research. Simulation allows an "as good as possible" approach to the modeling of systems, i.e., almost every imaginable detail of the system in the model can be captured in the simulation, e.g., the probability distributions for the processes can be arbitrary. Models of large scale systems tend to be very complex, so the coding and validation of the model can be very arduous.

The drawback for the high modeling power is the lack of any kind of methodological analysis. Furthermore, in order to obtain small confidence intervals, the simulation runs tend to be very long and expensive. The results of a simulation run have to be analyzed with caution, because we do not get exact performance measures, but only a (hopefully) close *estimate*. Simulation enjoys a high credibility and is most useful for the final stage of the system design for the fine tuning of the system and the evaluation of different scheduling policies, etc.

Often we are interested in finding the optimal allocation of resources. This involves analyzing the sensitivity of the system with respect to the change of the parameters we are interested in. In order to find the gradient at an operating point we usually use a finite difference approximation. Therefore, we require two runs: one with the nominal set of parameters, and a second one where the parameter of interest is slightly perturbed. On one hand we want to make the perturbation as small as possible in order to get the true gradient. But this involves long run times in order to get the small confidence intervals which are necessary. On the other

hand, we want a large increment in the parameter in order to have short run times, but in general this will not give us the true gradient.

The efficiency of simulation can be increased by *perturbation analysis* where it is possible to find the gradient with respect to the change of parameters based on one long simulation run [HC79, HC83, Sur89]. The parameters can be continuous (e.g., processor speed) or discrete (e.g., buffer size).

At the top of the hierarchy is building a prototype of the system which is the most "reliable" way of analyzing the performance of a system, but also the most expensive. Prototyping is done once the system has been designed and no further changes are expected.

1.2 Goal of this Research

The major problem for the use of GSPNs is the problem of state space explosion. The goal of this research is to move GSPNs up in the modeling hierarchy shown in Figure 1.1, so we can use Petri nets for models which previously could only be analyzed by simulation or heuristics due to the large state space.

In the study of QNs, efficient algorithms have been developed for the *approximate* analysis of QNs whose topology (or service rules) locally violates the product form assumption [LZGS84, Mar79, ABS84]. These QNs are called *"almost"* PFQNs. The approximate analysis is done in two steps:

1. In the *structural decomposition*, the QN is partitioned in such a way that the topology and routing probabilities can be interpreted as being from a PFQN.

2. During the approximate analysis, the system is analyzed either iteratively (Marie's method, [Mar79], response time preservation [ABS84]), or in a single step (flow equivalent aggregation [LZGS84]).

In this work, we study the problem of reducing the underlying state space of

the CTMC in order to allow the analysis of large systems.

The basic approach is to partition the GSPN by a single cut and substitute the subnets by simpler constructs resulting in *aggregated net systems*, thus reducing the state space. At the same time, the logical properties of the unaggregated parts of the net system are preserved.

We combine features from queueing theory and the results from the structural analysis of Petri nets. We start out from marked graphs (MGs), a concurrent decision free formalism provided with a powerful synchronization mechanism generalizing conventional QNs. For the structural decomposition of MGs, we make extensive use of the structural theory of MGs. In QNs, the characterization of the *workload* of PFQNs (or *"almost"* PFQNs) is usually not a problem, while in MGs this step is not obvious. Here we define the workload of a server as the maximum number of customers in its input queue. In Petri nets, places represent queues while timed transitions represent servers. Furthermore, in the decomposition phase, we want to preserve some marking properties of the aggregated systems.

Once the MG is partitioned, any of the available approximate techniques for QNs (Marie's method, FEA) or delay equivalence [LW91, WL91] can be used. The latter is an iterative technique and was first introduced in the context of MGs. In addition, a new iterative technique, *response time approximation* (RTA), specifically designed for the case of a single cut is introduced, which shows good convergence properties. The four methods are discussed and the computational effort and accuracy discussed.

MGs are then generalized to *macroplace/macrotransition* nets (MPMT-nets), a new subclass of Petri nets based on the combination of the basic structures of state machines (SMs) and MGs. The practical modeling with this new subclass of net systems has some advantages in the field of manufacturing with respect to the use of the well known subclass of live and bounded free choice net systems, allowing

timeout and the limited sharing of resources.

The modeling power of MPMT nets is much higher than that of state machines (SMs) or MGs alone. RTA can therefore be seen as a more general approximation technique.

The work is organized as follows: In Chapter 2 we briefly review some of the pertinent literature of GSPNs and QNs. In Chapter 3 we establish the Petri net notation and review some of the important properties of PNs. In Chapter 4 we present the structural decomposition of MGs and a new iterative approximation technique (RTA) as a *hierarchical* decomposition approach Petri nets based on RTA. In Chapter 5 we present other approximation technique for QNs in the context of Petri nets and a recently introduced approximation technique for MGs. In Chapter 6, we present numerical results for MGs. In Chapter 7, a new subclass (MPMT-nets) of Petri nets is introduced, and several numerical examples are given in Chapter 8. Chapter 9 introduces SIMO cuts for MPMT-nets. In Chapter 10, a dual arm robotics testbed is examined. Conclusions are given in Chapter 11.

CHAPTER 2
Literature Review

2.1 Introduction

This chapter discusses the pertinent literature of QNs and GSPNs. First in Section 2.2 we give a brief review about Jackson and Gordon-Newell QNs. In Section 2.3 four general approximation techniques for QNs are briefly reviewed.

We then concentrate on efforts to contain the problem of state explosion in GSPNs. A very efficient computational approach, where bounds of the throughput can be found in polynomial time based on the net description, is presented in Section 2.4.1. In Section 2.4.2 the efficient construction of the underlying CTMC is examined. In Section 2.4.3 an iterative solution method for weakly interacting subnets is discussed. Finally in Section 2.5 results of an approximation method based on the combination of QNs and GSPNs is reviewed. The iterative method for the approximate solution of stochastic marked graphs introduced by Woodside [LW91, WL91] et al. as well as Marie's method will be discussed in detail later in this work once the structural properties of the decomposition have been defined.

2.2 Jackson Queuing Networks

Jackson [Jac57, Jac63] considered an arbitrary network of queues with N nodes, each consisting of m_i exponential servers with rate μ_i. The *state* of the system is denoted by the vector $\bar{k} = (k_1, k_2 \ldots k_N)$, where k_i is the number of customers in the i-th service center i and $K = \sum_{i=1}^{N} k_i$ is the total number of customers in the system. The outside arrival of customers to the system is modeled by a Poisson process. An example of an open Jackson queueing network is shown in Figure 2.1.

Let λ be the outside arrival rate of customers to the QN. The routing of customers through the QN is modeled as a random walk through the service centers. The customers leave the QN with probability one after a finite time. The routing is described by the matrix $R = r(i,j), i = 0 \ldots N,\ j = 1 \ldots N+1$.

$r(0,j)$ is the probability that center j will be first on a routing,

$r(i, N+1)$ is the probability that if center i is l^{th} on a routing then it is the last one,

$r(i,j)$ is the probability that if center i is l^{th} on a routing, then there is an $(l+1)^{th}$ element and it is center j.

The matrix R defines the random walk of the customer through the system. For the sake of simplicity, let $m_i = 1,\ i = 1 \ldots N$. The traffic flow equations (flow into a service center = flow out of a service center) are given by:

$$T_l = \lambda r(0,l) + \sum_{j=1}^{N} r(j,l) T_j \tag{2.1}$$

where T_l is the total throughput of costumers at service center l. Jackson proved that the steady state solution $p(\bar{k})$ is given by:

$$p(k_1, k_2, \ldots k_N) = p_1(k_1) p_2(k_2) \cdots p_N(k_N) \tag{2.2}$$

where $p_i(k_i)$ is the solution to the classical $M/M/m$ queueing system, i.e., the probability of finding k_i costumers in the queue at service station i is.

$$p_i(k_i) = C_i \rho_i^{k_i} \tag{2.3}$$

C_i is a normalizing constant chosen such that $\sum_{a=0}^{\infty} p_i(a) = 1$, $\rho_i = T_i/\mu_i$. If $r(i, N+1) = 0$ and $\lambda = 0$ no outside arrivals are permitted and the number of customers in the system is constant. Such a queuing network is known as a Gordon-Newell queuing network [GN67]. The structural equivalent of Gordon-Newell QNs in Petri

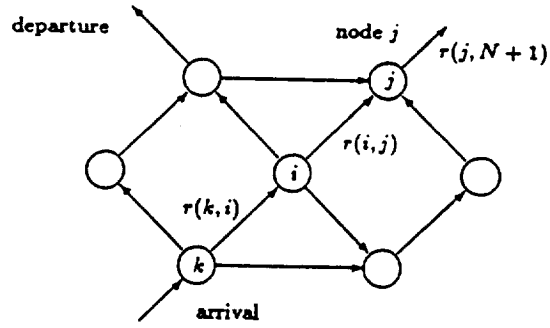


Figure 2.1: Example of Jackson Queuing Network

nets are state machines. The class of PFQNs was extended by the work of Baskett, Chandy, Muntz and Palacios [BCMP75]. The BCMP QN allows several classes of customers in the QN and different scheduling disciplines. The BCMP model has been expanded to include batch arrival and batch service in some special cases [HPTD90, HT90].

As soon as synchronization, limited buffer size, or the sharing of resources is present, a product form no longer exists. If only a "few" service centers violate the product form assumption, we loosely say the QN has *"almost"* product form.

2.3 Aggregation in Queueing Networks

All of the methods presented in the following are concerned with finding the equivalent service rate of the subnetworks to be aggregated. The service rate can be state dependent (i.e., dependent on the number of customers in the subnetwork) or state independent. See Table 2.1 for a comparison.

The basic approach is to set the conditional throughput (obtained under a controlled environment) to the service rate of the subnetwork, which in turn is substituted back in the original system.

Method	Arrival Process	Service Rate	Solution Method
Flow Equivalent Aggregation	short circuit, fixed number of customers	state dependent	single step
Response Time Preservation	state independent Poisson process	state independent	iterative
Marie's Method	state dependent	state dependent	iterative

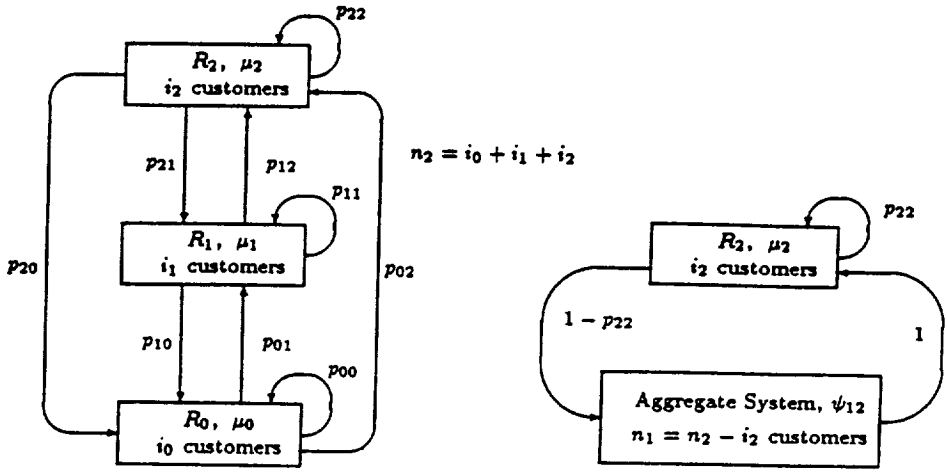
Table 2.1: Comparison of Approximate Methods for "Almost" PFQNs

2.3.1 Near Decomposability

If the order of the transition probabilities $\mu_0 p_{01}$ and $\mu_1 p_{10}$ between R_0 and R_1 in Figure 2.2.a is much greater than the order of magnitude between the transitions of R_2, then the QN formed by R_0 and R_1 will obtain local equilibrium before any interaction with R_2 can occur [Cou75]. A system with this property is known as nearly decomposable. The network can (approximately) be analyzed the following way: R_0 and R_1 are analyzed separately. We are interested in the flow $\psi_{12}(n_1)$ of customers out of the subsystem, where n_1 is the number of customers in R_0 and R_1. An aggregate model can be constructed which is shown in Figure 2.2.b. This approach was applied to GSPNs in the work of [BT86].

2.3.2 Flow Equivalent Aggregation (FEA)

The basic approach is to replace a general server or subnetwork of queues by a flow equivalent service center (FESC) [Bra85, LZGS84]. An arriving customer sees the FESC as a black box whose behavior is completely characterized by a listing of the residence time (the inverse of the throughput) as a function of the possible customer population. In order to determine the state dependent service rates, the subsystem is studied offline, i.e., without any interaction with the environment. In general, a FESC can only match the first moment of the probability distribution,



(a) System (b) Aggregated System

Figure 2.2: Nearly Decomposable System

as the higher moments are too cumbersome to obtain. Three steps for constructing an FESC can be determined [CS78]:

1. Determine a subsystem which can be analyzed by FESCs.

2. Analyze the subsystem by maintaining the number of customers constant. This is done by shorting out (Norton's theorem) all other parts of the network and varying the number of customers up to the maximum allowed in the subsystem.

3. The aggregated network AN is constructed as a server with a queue-dependent service rate. Let $\mathcal{X}_{SW}(k)$ be the conditional throughput of the subnetwork (SW) when k customers are present, and $\mu_{AN}(k)$ the (state dependent) service rate of AN. The approximation is done through the following equality, under the assumption of exponential service time:

$$\mathcal{X}_{SW}(k) = \mu_{AN}(k) \tag{2.4}$$

Figure 2.3 illustrates the concept from an example of computer systems [LZGS84]. Here the set of queues surrounded by the dashed box would be analyzed in isolation

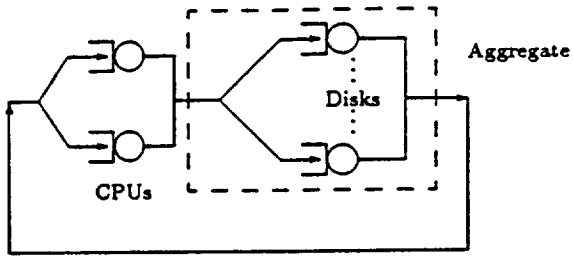


Figure 2.3: Concept of Flow Equivalent Server

and substituted by a single queue with a state dependent service time.

In FEA we assume that the behavior of the subnet is *independent* of the arrival process and depends only on the number of customers in the system, i.e., the behavior is completely independent of the *environment.* This condition is violated in several cases (e.g., if internal loops are present). Thus in general, the *mean completion time* (MCT) depends on the interarrival time of customers. FEA does not always lead to an approximate solution, as the following property shows.

Property 2.3.1 (Flow Equivalent Aggregation [Van78]) *For the particular case of product form QNs, flow equivalent aggregation leads to exact results.*

This approach does not involve any iterative process and is computationally efficient. As we do not iterate there exists no convergence problem. Courtois [Cou77] showed that the approximation is good when the number of state changes within the subsystem between interactions with the environment is reasonably large. Chandy et al. [CS78] observed that if the actual coefficient of variation of the service time is much larger than the coefficient of variation of the exponential distribution, significant error can be introduced. FEA techniques have been introduced in the context of SPNs [JD91a, JD91b] and are presented in Section 5.2.

2.3.3 Marie's Method

Marie's method provides an alternative way to derive the state dependent service rate of subsystems [Mar79] in an iterative fashion. While in FEA the number

of customers in the system is held constant, and in RTP the arrival rate is held constant, in Marie's method the arrival *and* service rate is assumed to be state dependent and must be found in an iterative manner. While the network may possess general service times, blocking and fork/join operations, the QN must be decomposable in the sense that the partitioned subnetworks can be interpreted as servers in a PFQN (i.e., the QN is in *"almost"* product form). To this end, the QN is partitioned into M subsystems according to the following definition:

Definition 2.3.1 (Product Form Decomposition) [BD90] *A non product form QN is said to be* decomposable *into product form if:*

1. *The transitions between subsystems are single transitions, e.g., bulk service or arrival between subnetworks is not allowed.*
2. *The behavior of the subnetwork depends only on its internal state, e.g., the service rate cannot depend on the queue length of another subnetwork.*
3. *The input/output behavior of a customer is independent of routing decisions taken during its stay in the subsystem, i.e., the* visit ratio *is unique.*
4. *All fork/join operations happen within the subsystem.*

After the partitioning, each subsystem is iteratively analyzed in isolation under a state dependent Poisson arrival process. As with many iterative methods, the uniqueness of the solution can not be proven although numerical experience has shown that a unique fixed point does indeed exist, although convergence sometimes presents a problem [BD90].

In [BD90], an introduction and comparison between Marie's method and FEA is given. The authors conclude that the results obtained using Marie's method are usually closer to the actual solution than those obtained with FEA, especially when large coefficients of variation are encountered. In terms of computational effort,

Marie's method seems to be more efficient for large values of N, while FEA is more efficient for small values of N.

A detailed discussion of the numerical aspects of Marie's method and its application to GSPNs is presented in Section 5.3.

2.3.4 Response Time Preservation

In response time preservation (RTP) [ABS84], the QN is first decomposed according to the rules in definition 2.3.1. In the stochastic phase, we replicate the response time of a general server G with first come, first served (FCFS) discipline by an exponential server as shown in Figure 2.4. Here the assumption for the analysis phase is that the complement of the general server (shown in the dashed box in Figure 2.4.a) generates a Poisson arrival rate μ of customers. The isolated subsystem shown at the bottom of Figure 2.4.a is the classical $M/G/1$ network of two queues: a single exponential server of rate μ and a complex server with general service time G. Let s be the service time, CV be the coefficient of variation and $\mathcal{X}$ be the throughput of the general server (which could also be formed by a subnetwork). By the Pollazcek-Khinchin formula [Kle75], the response time for an open $M/G/1$ queue is given by:

$$RT_G = s + \frac{(1 + CV^2)s \cdot U}{2(1 - U)} \tag{2.5}$$

where $U = \mathcal{X}s$ is the server utilization. During the aggregation phase RT_G is replicated by the response time RT' of the isolated exponential server as shown in Figure 2.4.b.

$$RT_G(\mu) = RT'(\mu) \tag{2.6}$$

RT' for an open $M/M/1$ with service time s' is given by:

$$RT' = \frac{s'}{1 - \mathcal{X}s'} \tag{2.7}$$

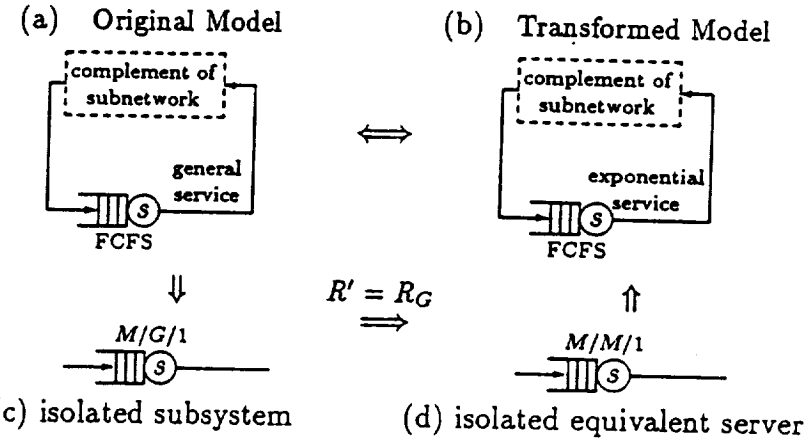


Figure 2.4: RTP for FCFS Server with General Service Time

Solving equation (2.5) and (2.7) for the unknown service time s' yields

$$s' = \frac{RT_G}{1 + \mathcal{X} RT_G} \tag{2.8}$$

In the last step the exponential server is substituted back into the original system to yield the transformed model which is now in product form. Ideally we select μ such that the $M(\mu)/G/1$ queue has the same response time as the system in Figure 2.4.d. As we do not know the throughput *a priori*, we also cannot find the correct Poisson arrival at the beginning. An iterative process is therefore concerned with finding μ. Note that in RTP, the coefficient of variation of the service time is required. RTP yields good results for moderate coefficients of variations ($CV \leq 5$) of the customer interarrival time [ABS84].

Iterative methods are especially suitable when the interaction of the subnet with the environment is strong. The drawback is that convergence might eventually present a problem.

2.4 Approaches to Contain the Problem of State Explosion in Generalized Stochastic Petri Nets

2.4.1 Bounds on the Throughput Vector

Whenever possible it is desirable to base the analysis of a PN on algorithms which are of polynomial complexity on the net structure rather than on the reachability graph (whose construction is of exponential complexity). J. Campos et al. use a linear program based on the incidence matrix to find upper and lower bounds for the throughput of MGs [CCCS92]. These bounds are tight, i.e., there exist probability distributions which will reach the bounds. The results have been generalized to Petri nets with a unique repetitive firing count vector [CCS91a] and free choice nets [CCS91b, Cam91]. Some of the results are presented in Section 3.4.2.

2.4.2 Efficient Construction of the Tangible State Space

Tangible markings correspond to states where the system spends some amount of time, while vanishing states often correspond to logical states and represent some intermediate decision process. Typically during the construction of the reachability graph, quite a few number of vanishing markings are encountered [BCR87]. In the solution of GSPNs we are more constrained by memory than time (at least when using the software package SPNP [DBCT85]), so much could be gained by using a technique which avoids the construction of the entire state space, and would reduce the number of vanishing states which have to be considered when reducing the CTMC to the set of tangible markings.

In [BCFR87] G. Balbo et al. examine the problem of the efficient construction of the tangible reachability graph. The reduction is done at a structural level, partitioning the GSPN into confusion free subnets. The reachability graph of each subnet is generated and the set of tangible states can be found.

2.4.3 Iterative Method for the Solution of Generalized Stochastic Petri Nets

In [CT91b, CT91a] Gianfranco Ciardo et al., propose an iterative (approximate) method for the solution of GSPN models. They identify several Petri net constructs which interact weakly. The solution of the GSPN is obtained by iteratively solving the subnets. In addition to *tight synchronization* (*parbegin-parend*), the following constructs were identified:

- **Processor-Sharing:** In this construct, two disconnected subnets share an implicit resource, i.e., a resource which is not explicitly represented by a place. Now the firing rates of two subnets are mutually dependent on each other depending on the amount of resources the other subnet has already used.

- **One-way condition testing:** Figure 2.5 shows a GSPN with the one way condition testing. Transition t_1 is enabled only if place p_1 is marked, but its firing does not remove a token from p_1. As Subnet SN_2 can be solved in isolation, and the results are going to be exact. In order to solve for the subnet SN_2, two methods have been proposed:

  1. *Rarefaction:* The argument here is that the absence of the marking of p_1 can only slow down transition t_1. The equivalent firing rate for t_1 can then be computed as:
     $\lambda_{t_1}^{new} = \lambda_{t_1}^{old} \mathrm{P}\{p_1 \text{ is marked}\}$

  2. *Compression:* Here the idea is to compress the behavior of subnet SN_1 into a small subnet SN_3. Now SN_2 and SN_3 can be combined for the analysis of SN_2. The analysis is going to be exact if the time in which p_1 is marked is exponentially distributed.

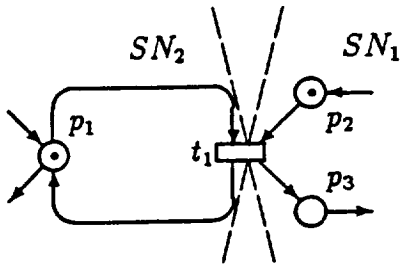


Figure 2.5: One Way Condition Testing in a Petri net

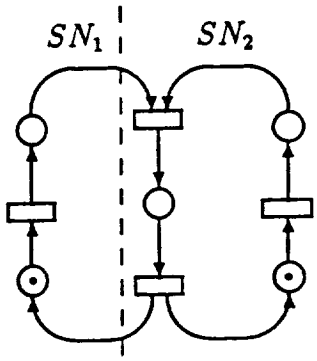


Figure 2.6: Producer and Consumer

- **Producer and Consumer**

  The construct is shown in Figure 2.6. This construct represents a synchronization between the two processes of net SN_1 and SN_2.

2.5 Approximate Aggregation in Generalized Stochastic Petri Nets

G. Balbo et al. combined QNs with GSPNs for the computationally efficient analysis of complex systems [BBG86, BBG88]. They combine features from QNs and GSPNs for an efficient analysis. In [BBG86] they start from a GSPN model. They then identify subnets which allow for a product form solution, and find flow equivalents for these subnets. These subnets are then substituted into a high level GSPN. This allows for a reduction of the state space.

The authors use a hybrid modeling approach. Parts of the system are modeled by QNs (either directly in QN notation or simply by state machines), while other parts are modeled by GSPNs. If the QN has a product form, then the flow equivalent server can be found analytically, by solving the PFQN. If the QN does not have a

product form, then the QN has to be solved (for example by means of a GSPN), and a flow equivalent server constructed. In this case the flow equivalent server has the form of a timed transition with a marking dependent firing rate.

In both papers, Balbo et al. use QNs as the underlying model, and only use GSPNs whenever necessary, to represent parts which cannot be represented by QNs. In the proposed work, we want to use GSPNs as the underlying model and only use the techniques adapted from QNs for the analysis, rather than QNs themselves.

2.6 Summary

In this chapter we have briefly reviewed some of the relevant literature for QNs. Of special interest are FEA, RTP and Marie's method as they are well known methods for the approximate analysis of QNs. Later in this work, we will use these techniques to devise efficient approximation techniques for MGs.

An approximation method for MGs by Li et al. will be reviewed once the structural decomposition of MGs has been defined.

CHAPTER 3
Terminology and Basic Properties for Petri Nets

3.1 Introduction

In this section the basic definitions for Petri nets are introduced. It is assumed that the reader is familiar with the basic concepts of Petri nets. See [Pet81, Mur89, Sil85] for an introduction.

3.2 Terminology

Definition 3.2.1 (Petri net) *A* Petri net structure *is a bipartite directed graph represented by a quadruple* $\mathcal{N} = \langle P, T, Pre, Post \rangle$ *where,*

$P = \{p_1, ..., p_n\}$	*is a finite set of* places, $n = \|P\|$
$T = \{t_1, ..., t_m\}$	*is a finite set of* transitions $m = \|T\|$
$Pre : P \times T \mapsto \mathbb{N}$	*is an input function that defines the set of directed arcs from* P *to* T
$Post : T \times P \mapsto \mathbb{N}$	*is an output function that defines the set of directed arcs from* T *to* P

$\mathbb{N}$ *is the set of nonnegative integers.* $P \bigcap T = \emptyset$, $P \cup T \neq \emptyset$.

Here, Pre and $Post$ are linear incidence functions representing the input and output arcs respectively.

Alternatively, a graph-based perspective of the net structure is given by $\mathcal{N} = (P, T, F, W)$ where F is a *flow relation*, $F \subseteq (P \times T) \cup (T \times P)$, and W is a weighting function on F. If the net is ordinary (all weights are 1), W can be removed (i.e., an *ordinary net* $\mathcal{N}$ can be defined as $\mathcal{N} = (P, T, F)$). Here we consider only ordinary nets.

Definition 3.2.2 (Incidence Matrix) *The* incidence matrix *of the net structure $\mathcal{N}$ is the $n \times m$ matrix C, where $C = Post - Pre$. $Post$, Pre and C represent matrices, while $Post(t_i)$, $Pre(t_i)$ and $C(t_i)$ represent the respective column vectors corresponding to transition t_i, while $l(p_i)$ represents the row vector of p_i in C.*

Definition 3.2.3 (Marking) *A* marked Petri net *$(\mathcal{N}, M_0)$, is a net provided with an initial marking M_0. $M(p)$ refers to the number of tokens residing in p.*

In this thesis *net* refers to the structure, while *net system* refers to the structure plus some initial marking.

Definition 3.2.4 (Firing of Transitions) *Transition t is firable (enabled) at marking M, if $M \geq Pre(t)$. If the firing of t leads to M' (i.e., $M[t\rangle M'$) then*

$$M' = M + Post(t) - Pre(t) = M + C(t) \tag{3.1}$$

Equation (3.1) is referred to as the linear state equation. Assuming $\sigma = t_{i1}t_{i2}\ldots t_{ir}$ is a sequence of transitions firable from M_0 (i.e., $M_0[\sigma\rangle M$), then $M = M_0 + C \cdot \vec{\sigma}$, where $\vec{\sigma}$ is the $m \times 1$ firing count vector (i.e., the firing order is lost). If $\vec{\sigma}$ is a vector, then $\vec{\sigma}(j)$ is the scalar representing the number of firings of t_j in σ.

The pre-set and post-set of a transition and place are defined as follows:

Definition 3.2.5 (pre-set, post-set)

${}^\bullet t = \{p | Pre(p,t) \geq 1\}$ = pre-set of t, i.e., the set of input places of t

$t^\bullet = \{p | Post(t,p) \geq 1\}$ = post-set of t, i.e., the set of output places of t

${}^\bullet p = \{t | Pre(p,t) \geq 1\}$ = pre-set of p, i.e., the set of input transitions of p

$p^\bullet = \{t | Post(t,p) \geq 1\}$ = post-set of p, i.e., the set of output transitions of p

Definition 3.2.6 (reachability set) *M_n is said to be reachable from the initial marking iff there exists a firing sequence σ that transforms M_0 into M_n. The* reachability set *$R(N, M_0)$ is the set of markings which are reachable from the initial marking M_0. A marking M is* potentially reachable *iff* $\exists \vec{X} \geq 0 \ni M = M_0 + C \cdot \vec{X} \geq 0$

Definition 3.2.7 (home state [Cam90]) *A state M is a home state iff $\forall M' \in R(N, M_0)$, $M \in R(N, M')$. Home states form a closed (i.e., mutually reachable) set.*

Later we will see, that the presence of home states leads to ergodic systems.

Definition 3.2.8 (Marking Bound) *Let $\langle \mathcal{N}, M_0 \rangle$ be a net system*

1. *The* bound *of a place $p \in P$ is*

$$B(p) = \max\{M'(p) \mid M' \in R\langle \mathcal{N}, M_0 \rangle\} \tag{3.2}$$

   where $M'(p)$ refers to the number of tokens in place p.

2. *The structural bound of a given place p of $\mathcal{N}$ is given by the following linear program:*

$$\begin{aligned} SB(p) \stackrel{\text{def}}{=} \quad & \textit{maximize} \quad M(p) \\ & \textit{subject to} \quad M = M_0 + C \cdot \vec{\sigma} \geq 0 \\ & \qquad\qquad\quad \vec{\sigma} \geq 0 \end{aligned} \tag{LPP1}$$

A net $\mathcal{N}$ is structurally bounded *iff $\forall M_0$ the net is bounded.*

i.e., all net systems constructed from $\mathcal{N}$ are bounded.

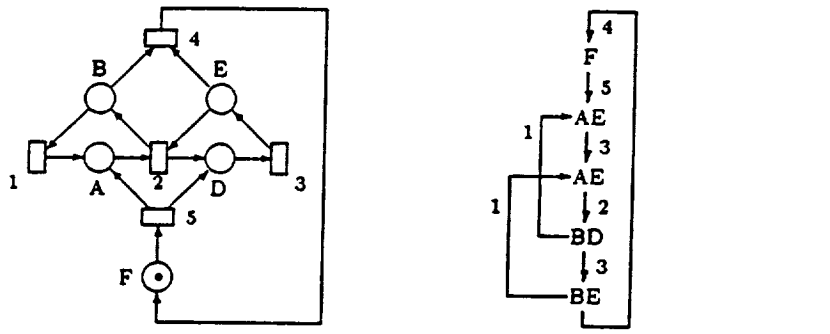
Definition 3.2.9 (persistent place) *A place is persistent iff $|p^\bullet| = 1$*

Once a persistent place is marked, the routing of the token out of the place is deterministic.

Definition 3.2.10 (Liveness) *A transition $t \in T$ in $\langle N, M_0 \rangle$ is live if*

$$\forall M \in R(N, M_0) : \exists M' \in R(N, M) \ni M' \geq Pre(t)$$

$\langle N, M_0 \rangle$ is live iff all transitions are live.



(a) Petri net system

(b) Reachability graph (Marked places are indicated by letters)

Figure 3.1: Ordinary Petri Net System and its Reachability Graph

Definition 3.2.11 (Invariants [Cam90]) *A* T-invariant *(or consistent component) is an* m-*vector* $X : T \mapsto I\!N$ *such that* $C \cdot X = 0$ *and* $X \neq 0$. *A* P-*invariant (or conservative component) is a* n-*vector* $Y : P \mapsto I\!N$ *such that* $Y^T \cdot C = 0$ *and* $Y \neq 0$. *The support of a* T-*invariant (*P-*invariant) denoted by* $\|X\|$ *(*$\|\mathcal{Y}\|$*), is the set of transitions (places) such that* $\|X\| = \{t \in T | X(t) > 0\}$ *(*$\|Y\| = \{p \in P | Y(p) > 0\}$*). A (*$P$ *or* T*) invariant* I *is:*

1. Minimal *iff there exists no other invariant* I' *such that* $\|I'\| \subset \|I\|$.
2. Canonical *iff the greatest common divisor is 1.*
3. Elementary *iff it is minimal and canonical.*

The nonzero solution to $C \cdot X = 0$ ($Y^T \cdot C = 0$) is also called the T-semiflow (P-semiflow).

A firing count vector which leads back to the original marking is a T-invariant (but not necessarily vice versa). A net is *consistent* iff all transitions are covered by T-invariants, i.e., $\exists$ a T-invariant $X \ni X \geq \mathbb{1}$ ($\mathbb{1}$ is a vector of appropriate size whose entries are all one). In the support of a P-invariant, the weighted sum of tokens over a set of places is constant for any marking.

Figure 3.1 shows an example of a simple Petri net. For the given initial marking, there are five possible states (all states are home states). The incidence matrix is given by:

$$C = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 \end{matrix} \\ \left[\begin{array}{rrrrr} 1 & -1 & 0 & 0 & 1 \\ -1 & 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & -1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{array}\right] \end{array} \begin{matrix} A \\ B \\ D \\ E \\ F \end{matrix} \tag{3.3}$$

Solving for the invariants yields two minimal P-invariants:

$$Y_1^T = [1 \quad 1 \quad 0 \quad 0 \quad 1], \; Y_2^T = [0 \quad 0 \quad 1 \quad 1 \quad 1]$$

Two minimal T-invariants exist:

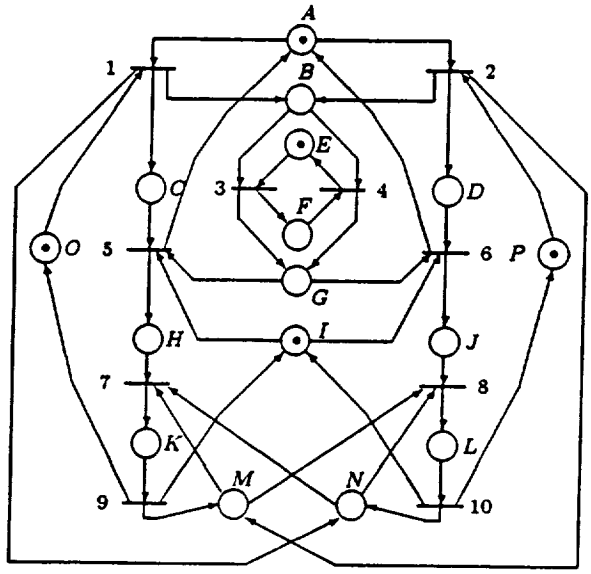
$$X_1^T = [1 \quad 1 \quad 1 \quad 0 \quad 0]^T, \; X_2^T = [0 \quad 1 \quad 2 \quad 1 \quad 1]^T$$

Consistency and boundedness do not guarantee other "nice" properties such as the existence of home states or even liveness. Figure 3.2 shows the example of a structurally live and structurally bounded Petri net system without home states and its reachability graph. Depending on whether t_1 or t_2 are fired first, the resulting states are disjoint and form closed sets. The net has 3 minimal T-invariants:

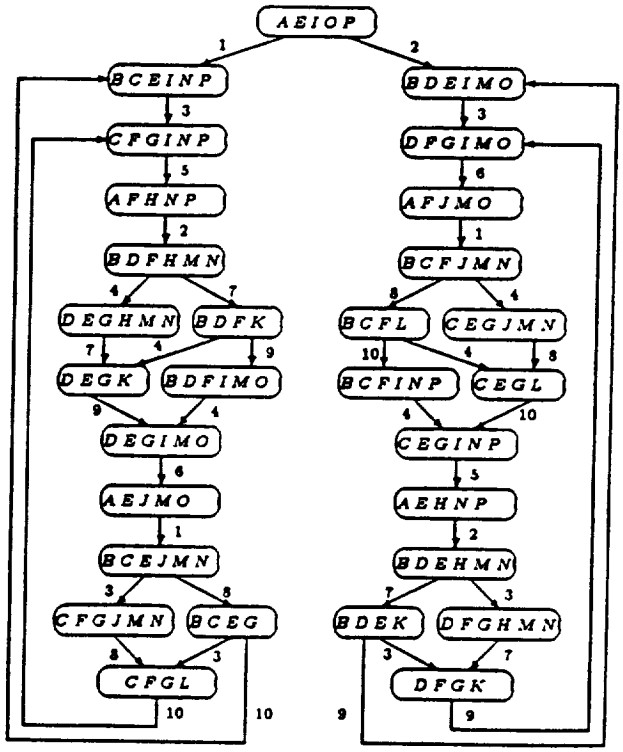
$$X_1^T = [1\,1\,1\,1\,1\,1\,1\,1\,1\,1]^T, \; X_2^T = [2\,0\,1\,1\,2\,0\,2\,0\,2\,2]^T, \; X_2^T = [0\,0\,1\,1\,0\,2\,0\,2\,0\,2]^T$$

For the given initial marking, only X_1^T leads to a feasible firing sequence, i.e., $\exists \sigma \ni \vec{\sigma} = X_1^T$. In general, the existence of home states depends on the initial marking!

Figure 3.3 shows a consistent, structurally live and bounded Petri net, which has an interesting property: By increasing the initial marking (e.g., $m_0(p_5) = 1$), the net system can deadlock.



(a) Live and Bounded Petri Net System



(b) and its Reachability Graph

Figure 3.2: Live and Structurally Bounded Petri Net System Without Home States

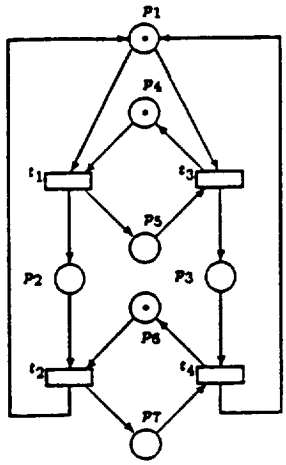


Figure 3.3: Structurally Live, Consistent and Bounded Petri Net System

3.3 Subnets

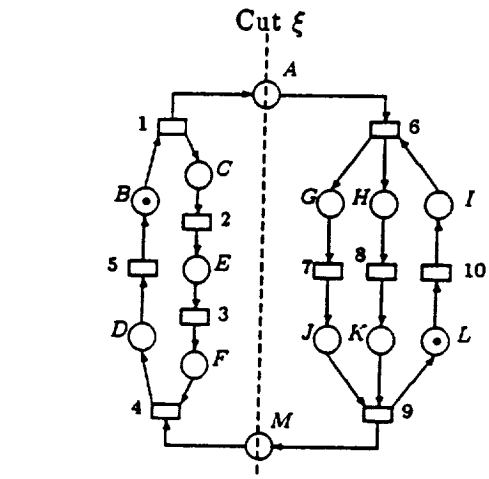
Definition 3.3.1 (Single Cut) *A single cut ξ traced through a set of interface places Q splits a net $\mathcal{N} = (P, T, F)$ into two subnets $SN_i, = (P_i, T_i, F_i)$ $i = 1, 2$, where*

- $T_1 \cup T_2 = T, \;\; T_1 \cap T_2 = \emptyset$
- $P_1 \cup P_2 = P, \;\; P_1 \cap P_2 = Q$ *{ the interface}*
- $F_i = F \cap ((P_i \times T_i) \cup (T_i \times P_i))$
- $Q = \{T_1^\bullet \cup^\bullet T_1\} \cap \{T_2^\bullet \cup^\bullet T_2\}$ *defining the interface between SN_1 and SN_2.*

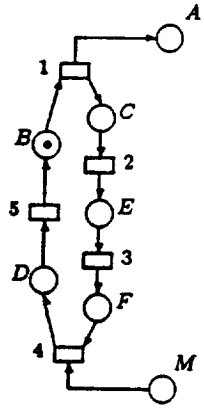
In other words, the subnet SN_i is characterized by T_i because $P_i =^\bullet T_i \cup T_i^\bullet$. The complement of SN_1 is SN_2 and vice versa. Figure 3.4 shows a single cut through a MG system and the corresponding subnets. The interface places are given by places A and M.

Definition 3.3.2 (Projection) *Let $\mathcal{N}$ be a Petri net and SN_i be a subnet of $\mathcal{N}$. Let $\overline{M}$ be a marking vector of $\mathcal{N}$. M_0^i is the projection of $\overline{M}$ on SN_i such that:*

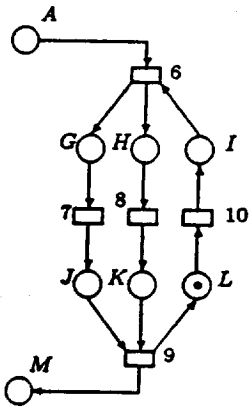
$$\forall p \in P_i, \quad M_0^i(p) = \overline{M}(p) \tag{3.4}$$



(a) Sample Petri Net N



(b) Subnet SN_1



(c) Subnet SN_2

Figure 3.4: Example for Single Cut Generating two Subnets

3.4 Petri Net Subclasses and Their Properties

In this section we introduce three well known subclasses of Petri nets (state machines, marked graphs and free choice nets).

Let us first introduce a well known property from graph theory.

Definition 3.4.1 (strongly connected [Mar71]) *A directed graph is said to be strongly connected iff a directed path exists between every pair of vertices.*

Therefore, a Petri net is strongly connected iff a directed path exists between any pair of places and transitions.

3.4.1 State Machines

Definition 3.4.2 (State Machine) *A* state machine *(SM) is a Petri net such that*

$$\forall t \in T \ \ |{}^\bullet t| = 1 \ \textit{and} \ |t^\bullet| = 1 \tag{3.5}$$

The following properties for state machines hold [SV90]:

Theorem 3.4.1 (State Machine)

i) SMs are structurally bounded and $Y = \mathbb{1}$.

ii) A SM is live iff $\mathcal{N}$ *is strongly connected and* $\sum_{i=i}^{n} M_0(i) \geq 1$.

From a structural point of view, closed monoclass PFQNs (Gordon-Newell QNs) can be represented by strongly connected SMs. SMs allow choices but cannot model synchronization.

3.4.2 Marked Graphs

Definition 3.4.3 (Marked Graphs) *A* marked graph *(MG) is an ordinary net such that any place has one input and one output transition.*

$$\forall p \in P \ \ |{}^\bullet p| = 1 \text{and} \ |p^\bullet| = 1 \tag{3.6}$$

MGs allow the modeling of concurrency and synchronization but not of decisions. From the perspective of Petri nets, the structure of *monoclass Fork/Join Queueing Networks (QNs) with Buffers* (FJQN/B) [DLT90] is represented by strongly connected MGs [DLT90].

For MGs, the properties stated in the following theorems can be established using polynomial time algorithms:

Theorem 3.4.2 [Mur89] *Let* $\mathcal{N}$, *be a MG:*

i) $(\mathcal{N}, M_0)$ *is live iff all circuits (P-invariants) are marked.*

ii) $\mathcal{N}$ is structurally bounded (i.e., bounded for any M_0) iff it is strongly connected.

Corollary 3.4.1 *MGs are structurally live*

Corollary 3.4.2 [Cam90] *The liveness of a MG can be checked by the absence of unmarked P-invariants.*

$$\nexists Y \gneq 0 \ni Y^T \cdot C = 0, Y^T \cdot M_0 = 0, \tag{3.7}$$

where $Y \gneq 0$ means $Y \geq 0$ and $Y \neq 0$.

Theorem 3.4.3 [Mur77] *Let $\langle \mathcal{N}, M_0 \rangle$ be a live MG system. The three following statements are equivalent:*

i) M is reachable from M_0 (i.e., $\exists \sigma$ such that $M_0[\sigma\rangle M$).

ii) M is an integer solution of $M = M_0 + C \cdot \vec{\sigma} \geq 0$, $\vec{\sigma} \geq 0$.

iii) M is an integer solution of $Y^T \cdot M = Y^T \cdot M_0$ for any P-invariant Y (i.e., $\exists Y \geq 0$, $Y^T \cdot C = 0$).

In MGs, all states (in particular M_0) are home states.

Corollary 3.4.3 *Let $\langle \mathcal{N}, M_0 \rangle$ be a live MG system, then $\forall p \in P : SB(p) = B(p)$.*

i.e., the structural and marking bound are the same.

Corollary 3.4.4 *In MGs, the marking bound is given by the minimum number of tokens in a directed circuit containing p.*

Starting from a place p, a directed circuit in a Petri net is obtained by traversing along the arcs and eventually returning to p. In MGs, the places in a directed circuit are P-invariants.

Stochastic MGs are obtained by associating time to transitions, i.e., a transition fires after it has been enabled for some time. Tight insensitive (i.e., for any probability distribution) cycle time lower bounds are known for stochastic MGs [CCCS89].

Theorem 3.4.4 [CCCS92] *The minimum mean interfiring time (i.e., cycle time) for a live and strongly connected stochastic MG system can be obtained by solving the following linear programming problem:*

$$\begin{aligned} \Gamma^{min} = \quad & \textit{maximize} \quad Y^T \cdot Pre \cdot \vec{s} \;\; \textit{w.r.t.}\; Y \\ & \textit{subject to} \quad Y^T \cdot C = 0 \\ & \qquad\qquad\;\; Y^T \cdot M_0 = 1 \\ & \qquad\qquad\;\; Y \geq 0 \end{aligned} \tag{LPP2}$$

where $\vec{s}$ is the vector of the mean service times of the transitions. Moreover, there exist probability density functions for which the bound can be reached (tightness). In the particular case of deterministic, timing the bound is reached.

According to the previous result, a lower bound for the mean interfiring time of a live and bounded MG system can be computed in *polynomial time* solving the linear programming problem (LPP2). From the theory of LPP [NRKT89], it is well known, that optimal solutions appear at the *extremal feasible solutions.* In (LPP2), feasible solutions are P-invariants (i.e., $Y \geq 0$, $Y^T \cdot C = 0$), and the *extremal* solutions are the *minimal* P-invariants. Moreover, minimal P-invariants define *elementary directed circuits* of the net (Theorem 3.4.3), while the constraint $Y^T \cdot M_0 = 1$ is just a normalization. Therefore, in order to deal with the interpretation of the computation in (LPP2) it is enough to observe that $Y^T \cdot Pre \cdot \vec{s}/Y^T \cdot M_0$ is nothing more but the sum of service times associated with the transitions belonging to the elementary circuit defined by the minimal P-invariant Y, divided by the number of tokens in that circuit. This value is just the *mean interfiring time of any transition of the circuit if it is assumed to work in isolation.* Therefore, (LPP2) computes the mean interfiring time of the *slowest* circuit considered in isolation. In other words, (LPP2) only gives a lower bound, because it disregards the interaction with other circuits (i.e., it disregards the waiting times due to the synchronization among

circuits). Later in this work we will use the theory of bounds to find suitable initial conditions for Response Time Approximation (RTA).

As a final remark, Γ^{min} (in LPP2) is finite iff there exist no unmarked P-semiflows (i.e., directed circuits). If Γ^{min} is unbounded, then a deadlock (thus non liveness) is detected. Because the bound given by (LPP2) is tight, liveness, deadlock-freeness and the absence of unmarked circuits for bounded MG systems are equivalent conditions.

Theorem 3.4.2 states that liveness and boundedness can be established on the net structure and initial marking. Theorem 3.4.3 states that every integer solution (e.g., obtained by solving a linear system of equations) is indeed a reachable state, while LPP1 (Definition 3.2.8) allows the computation of the marking bound in polynomial time [Kar84].

State Machines and *Marked Graphs* are *dual* net subclasses (i.e., by exchanging places with transitions, the structure of MGs is obtained from the structure of SMs and vice versa). The *reverse* of a net is obtained by reversing the flow relation (i.e., the arcs). The reverse of a MG (SM) is a MG (SM), in other words, MGs and SMs are self-reverse net subclasses. Therefore the reverse-dual of a MG (SM) is a SM (MG).

3.4.3 Free Choice nets

Free choice nets (FC-nets) are a well known subclass of Petri nets which can model sequence, choice and concurrency. In FC-nets, choices and synchronization do not directly interfere with each other, i.e., they cannot coincide on the same transition. More formally, a *FC-net* is an ordinary net such that if two transitions have a common input place, this is the only input place of both transitions.

Definition 3.4.4 (free choice nets [Hac72]) *Free choice nets are ordinary Petri nets such that*

$$\forall p \in P: \ |p^\bullet| > 1 \Rightarrow \ {}^\bullet(p^\bullet) = \{p\}$$

The following theorem can be used to determine whether a given structurally bounded FC-net is structurally live.

Theorem 3.4.5 [CCS91b] *Let C be the incidence matrix of the strongly connected structurally bounded FC-net $\mathcal{N}$. Then the net $\mathcal{N}$ is structurally live iff $rank(C) = m - 1 - (a - n)$, where $m = |T|$, $n = |P|$ and a is the total number of input arcs to the transitions.*

Theorem 3.4.6 [ES90b] *Let $\langle \mathcal{N}, M_0 \rangle$ be a live and bounded FC net, then*

$$\forall p \in P: \ \ B(p) = SB(p)$$

Similarly to MGs, we can find sufficient conditions for the liveness and boundedness of FC nets based on the *net structure* with polynomial time algorithms.

3.5 Stochastic Petri Nets

In order to use Petri nets for the performance analysis of systems, the notion of time has to be introduced. There are potentially two ways to incorporate time into a Petri net:

i) By associating time to places, i.e., places have to be marked for some time before the tokens can be removed [Ram74, WDF85],

ii) or by associating time to transitions. In the literature, two different firing rules have been defined:

   (a) In *two stage firing*, the transition absorbs the tokens and releases them after some time [Zub85],

(b) In the *one stage firing* rule, transitions fire after being enabled for some time, i.e., the firing is *atomic* and identical to the usual firing rule [Mol82, RP84].

By associating time to transitions, Petri net models directly represent synchronized QNs [Cam90], where places represent queues and transitions represent service stations or fork/join operations.

In any of the two models, the *conflict resolution* has to be addressed. In the *two stage firing* of transitions, the resolution of conflict is predetermined either deterministically or stochastically by associating a routing probability to the transitions in conflict. In the *atomic firing*, conflict is usually resolved by *race policy*, i.e., the transition which samples the minimum service time fires first. If other conflict resolutions are used (e.g., age memory) the models become increasingly complex [AMBC+89]. Stochastic Petri nets (SPNs) with *atomic* firing can model pre-emption, while this is not possible for SPNs with the two stage firing rule. Due to their larger modeling power, in the present work we adopt SPNs with time associated to transitions and atomic firing rule.

We assume that the delay associated with transition t_i is a nonnegative continuous random variable X_i with the exponential distribution function.

$$F_{X_i}(x) = \Pr[X_i \leq x] = 1 - \exp(-\lambda_i x) \tag{3.8}$$

i.e., the probability density function (pdf) is $f_{X_i} = \lambda_i \exp(-\lambda_i x)$.

The average delay is given by:

$$s_i = \int_0^\infty [1 - F_{X_i}(x)]\mathrm{d}x = \int_0^\infty \exp(-\lambda_i x)\mathrm{d}x = \frac{1}{\lambda_i} \tag{3.9}$$

λ_i is called the firing rate of t_i.

The exponential distribution is the only continuous time probability distribution which has the memoryless (Markovian) property,

$$P\{X \geq x + \alpha | X \geq \alpha\} = P\{X \geq x\} \tag{3.10}$$

It is possible to show [Mol82], that the reachability graph of a bounded SPN with exponential firing distribution is isomorphic to a CTMC.

Activities which do not require any amount of time are often modeled by logical states in the sequence of events and can be modeled by using *immediate transitions.* In the graphical representation, immediate transitions are drawn as thin black bars, while timed transitions are drawn as thick white bars. The probability that the immediate transition t_k from the enabled choice set $S(M)$ fires is:

$$P\{t_k\} = \frac{r_k}{\sum_{i:t_i \in S(M)} r_i} \tag{3.11}$$

where $r_i \in I\!N^+$ is the *probability rate* of t_i. Subsequently we have two types of markings:

- A *Vanishing marking* is a marking where an immediate transition is enabled. The GSPN leaves that state immediately. If there are no trapping loops consisting only of vanishing markings, the probability of finding the GSPN in that marking is zero. Vanishing markings occur frequently in the execution of the GSPN, and are often associated with logical states [BCR87]. For example, after a task is completed, a vision system will not remain idle if another task is waiting (it will leave the idle state immediately and become busy). In the construction of the CTMC the vanishing markings have to be eliminated.

- A *Tangible marking* is a marking where only timed transitions are enabled. The GSPN has a nonzero probability of being in that state. These states are associated with states where some sort of non-zero length activity takes place. A typical example would be the downloading of an image into the memory of the computer.

An important performance measure is the *visit ratio* of transitions. It can be defined as the normalized (with respect to a transition) throughput.

Definition 3.5.1 (visit ratio) *Let $\vec{\mathcal{X}}$ be the vector of throughputs in steady state and $\mathcal{X}_j$ its j-th entry (i.e., the throughput of t_j) of the net system $\langle \mathcal{N}, M_0 \rangle$. The vector of visit ratios $\vec{v}^{(j)}$ (normalized for t_j) is given by*

$$\vec{v}^{(j)} = \frac{1}{\mathcal{X}_j}\vec{\mathcal{X}} \tag{3.12}$$

If the conflicts in FC nets are modeled by immediate transitions, the visit ratios can be computed based on the net structure [Cam90]. It is obvious that $\vec{v}^{(j)}$ must be a T-invariant, i.e.,

$$C \cdot \vec{v}^{(j)} = 0 \tag{3.13}$$

Let t_{i1}, t_{i2} be two transitions in structural conflict (i.e., with the same pre-set) with probability rates r_{i1} and r_{i2} respectively and $\mathcal{X}_i$ the flow into ${}^\bullet t_{i1}$. Then the throughput of t_{i1} and t_{i2} is given by:

$$\mathcal{X}_{i1} = \frac{r_1}{r_1 + r_2}\mathcal{X}_i \tag{3.14}$$

$$\mathcal{X}_{i2} = \frac{r_2}{r_1 + r_2}\mathcal{X}_i \tag{3.15}$$

Dividing eq. (3.14) by eq. (3.15) and rearranging terms yields

$$r_{i1}v_{i2}^{(j)} = r_{i2}v_{i1}^{(j)} \tag{3.16}$$

By considering all conflicts in the net, eq. (3.16) can be combined together with eq. (3.13) and can be rewritten in matrix form as

$$\begin{pmatrix} C \\ R \end{pmatrix} \cdot \vec{v}^{(j)} = 0 \quad v_j^{(j)} = 1 \tag{3.17}$$

Eq. (3.17) has a unique solution iff the number of the independent rows of the matrix is $m-1$ with $m = |T|$. J. Campos [Cam90] showed that the number of independent rows in the matrix in eq. (3.17) for subclasses of Petri nets which include FC nets, is indeed $m-1$.

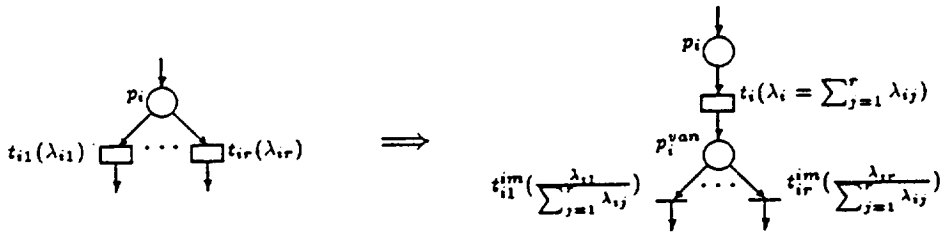


Figure 3.5: Modeling of Timed Structural Conflict with Immediate Transitions (Firing and Probability Rates are in Parentheses)

Consider the structural conflict modeled with timed transitions in Figure 3.5.a. Since the minimum of two exponential distributed variables X_1 and X_2 with rate λ_1 and λ_2, $\min(X_1, X_2)$ is again an exponential distributed random variable with rate $\lambda_{\min(X_1,X_2)} = \lambda_1 + \lambda_2$, the sojourn time of a token in p_i is a random variable with negative exponential pdf with mean

$$s_i = \frac{1}{\sum_{j=1}^{r} \lambda_{ij}} \tag{3.18}$$

i.e., the rate out of p_i is $\sum_{j=1}^{r} \lambda_{ij}$ [Mar89]. The probability that t_{ik} samples the minimum time (i.e., that it will fire) is

$$\Pr(t_{ik}) = \frac{\lambda_{ik}}{\sum_{j=1}^{r} \lambda_{ij}} \tag{3.19}$$

Based on the previous discussion we can transform Figure 3.5.a into an equivalent conflict with immediate transitions as shown in Figure 3.5.b. The steady state probability of place p_i^{van} being marked is zero, while the immediate transitions t_{ir}^{im} model the conflict.

Theorem 3.5.1 *Let $\mathcal{N}$ be a stochastic FC net and p_i be the input place of a structural conflict modeled with exponentially timed transitions. The transformation presented in Figure 3.5 preserves the underlying CTMC.*

Proof: It is a direct consequence of the construction rules for the underlying CTMC from the reachability set that includes vanishing markings. □

In our model we do not allow *priorities* to be associated with immediate transitions or inhibitor arcs. An alternate definition of the generalized stochastic Petri net (GSPN) defined in [Mar89] is therefore given by:

Definition 3.5.2 (GSPN) *The GSPN is a six-tuple*

$$GN = (P, T, Pre, Post, M_0, R) \tag{3.20}$$

$(P, T, Pre, Post, M_0)$ *is the underlying Petri net system.*

$R = (r_1, r_2 \ldots r_m)$ *is the array where* r_i

- *is the parameter of the exponential probability density function of the firing delay if* t_i *is a timed transition,*
- *is a weight used for the computation of firing probabilities of immediate transitions if* t_i *is an immediate transition.*

From a performance analysis point of view, we want to estimate the average behavior of the system in the long run. The usual assumption is that the system is *ergodic* [Ros83], i.e., in the long run the average of the observed values tend (almost surely) to the theoretically expected ones. In finite ergodic systems, if starting in state i, the expected time to return to state i is finite.

Definition 3.5.3 [Cam90] *A stochastic process* Z_τ *(*$\tau > 0$ *representing the time), is said to be ergodic iff the following condition holds:*

$$\lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau Z_u \, du = \lim_{\tau \to \infty} E[Z_\tau] < \infty \tag{3.21}$$

Not all live and bounded stochastic Petri nets are ergodic. Consider the net in Figure 3.2, here two closed subsets of states exist. The steady state marking will depend on the initial resolution of the conflict between t_1 and t_2. We can circumvent this difficulty with the following important results:

Theorem 3.5.2 [Cam90] *If a Markovian bounded marked net has a home state, then its marking process is ergodic.*

Starting from the reachability graph and its stochastic interpretation, we can find the underlying homogeneous irreducible CTMC.

3.6 Conclusions

The practical modeling power of Petri nets is constrained by subclasses, but they allow more powerful analysis techniques using fast (e.g., polynomial time) algorithms that avoid the state explosion problem. These algorithms are based on establishing a relationship between the behavior of the net system and its structure, and are parameterized by the initial marking, which plays an auxiliary role.

In the present work we limit ourselves to the analysis of subclasses of net systems whose marking process is ergodic (Definition 3.5.3) and where all states are home states. Thus, we only deal with systems whose underlying CTMC is irreducible (Theorem 3.5.2). Furthermore, we restrict our attention to subclasses of net systems where we can determine ergodicity of the marking process based on the *net structure* $\mathcal{N}$ and an initial marking M_0, considering only net subclasses where the reachability of a marking can be determined based on the state equation (Theorem 3.4.3) and thus all states are home states. From the above discussion, we now limit our attention to two net subclasses: SMs and MGs. Strongly connected SMs can trivially be solved because they have the structure of monoclass Gordon-Newell QNs and well known product form solutions can be applied. For MGs, no product form solution is known, therefore in the following chapter we concentrate our efforts on this particular net subclass. In Chapter 7, we consider MPMT-nets where we show that the above properties also hold.

CHAPTER 4

Structural Decomposition and Response Time Approximation for Marked Graphs

4.1 Introduction

The basic approach in the presented approximation methods is to divide and conquer. We can identify two phases in the process: a *structural decomposition phase* (divide) and a *stochastic aggregation phase* (conquer). The structural decomposition phase consists of two steps:

1. Structural: the system is partitioned into subsystems yielding the aggregated nets $\mathcal{AN}$.

2. Marking: the marking of the aggregated nets is computed such that a notion of environment is kept (yielding the aggregated net *systems* $\mathcal{AS}$).

Based on the results of the decomposition phase, in the second phase the aggregated net systems subsystems are analyzed some assumptions which depend on the method we are using (e.g., arrival process, number of customers).

In the following we are concerned with finding suitable assumptions (arrival process) for the analysis phase, while at the same time preserving some information about the environment of the subsystem. For MGs with deterministic timing or Jackson QNs, the aggregation preserves the performance measures, but in general the representation of the environment presents a dilemma. *If we represent the complement of the subsystem better, the state space of the underlying CTMC is going to be larger.* In the proposed approach, we try to preserve the response time (RT) of the subsystems exactly. As we will see, this is only possible for certain special cases,

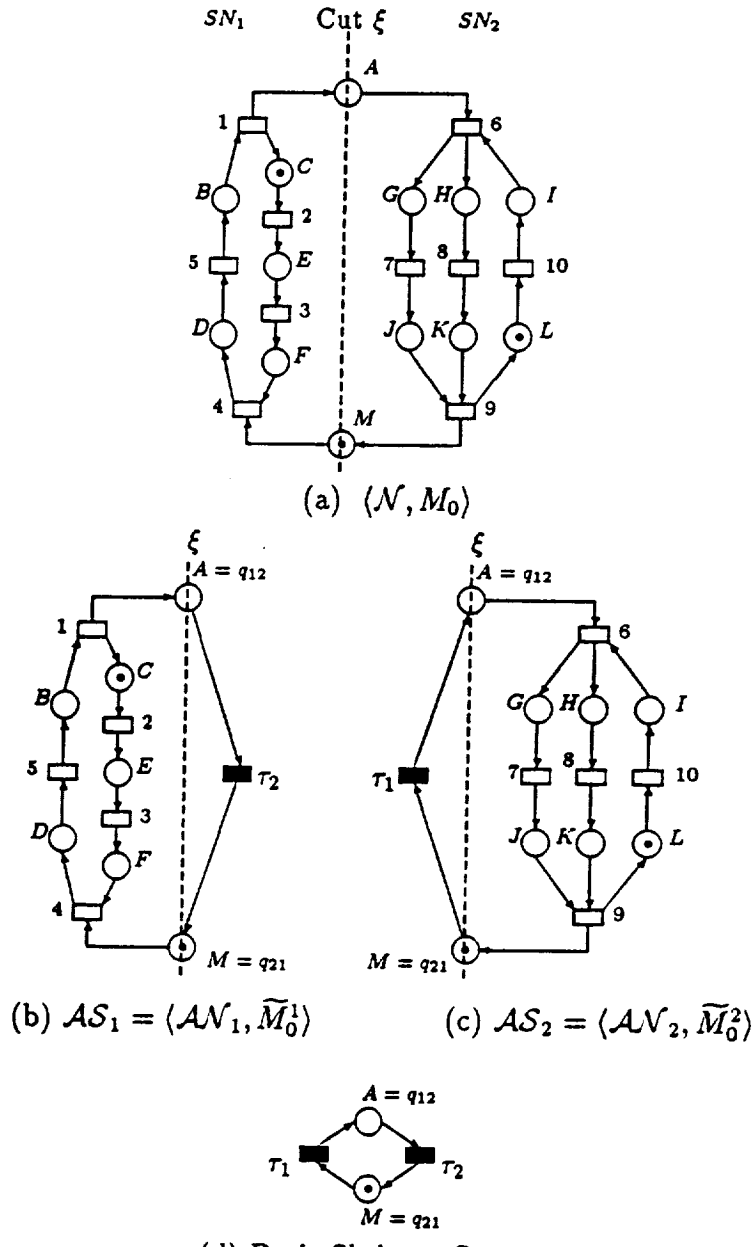
while in general we only *approximate* the RT. Therefore, the proposed method can be seen as an *RT approximation* (RTA) [JSS92b].

In this chapter, we present mostly the theoretical aspects of the decomposition. Several examples are introduced and analyzed in Chapter 6, after all of the aggregation techniques have been presented (Chapter 6).

4.2 Single Cut, Aggregated nets, Basic Skeleton and Response Time Approximation

The basic idea is illustrated in Figure 4.1.a: the net is split into two subnets by a single cut ξ defined through some places called *interface places* (A and M in Figure 4.1.a) [JSS92b, JSS92a]. From the cut we define three nets: two *aggregated nets* $\mathcal{AN}$ (one per subnet, Figures 4.1.b, $\mathcal{AN}_1$, and 4.1.c, $\mathcal{AN}_2$), and the *basic skeleton* (Figure 4.1.d). $\mathcal{AN}_i$ is obtained by preserving subnet SN_i and connecting the interface places through a transition τ_j representing the reduction of the other subnet $SN_j, j \neq i$. Transitions τ_1 (Figure 4.1.c) and τ_2 (Figure 4.1.b) represent the aggregation of subnets SN_1 and SN_2, respectively. The basic skeleton (i.e., the fully aggregated net, Figure 4.1.d) is obtained by maintaining the interface places, while reducing each subnet to a single transition. For a single cut, the basic skeleton always has the same structure, i.e., it is independent of the topology of the stochastic MG under analysis. The markings of $\mathcal{AN}_1, \mathcal{AN}_2$ (leading to the aggregated net system $\mathcal{AS}_i = \langle \mathcal{AN}_i, M_0^i \rangle$) and the basic skeleton are computed based on the initial marking of the original net.

Assuming *single server semantics* for the aggregated transitions τ representing the subnets, the basic skeleton will be interpreted as a classical $M/M/1/K$ queue, where K represents the number of customers in the system (and therefore also the maximum queue size). Therefore, the throughput $\mathcal{X}$ of the basic skeleton can be



(a) $\langle \mathcal{N}, M_0 \rangle$

(b) $\mathcal{AS}_1 = \langle \mathcal{AN}_1, \widetilde{M}_0^1 \rangle$

(c) $\mathcal{AS}_2 = \langle \mathcal{AN}_2, \widetilde{M}_0^2 \rangle$

(d) Basic Skeleton System

Figure 4.1: Aggregated Nets (b+c) and Basic Skeleton (d) Systems Obtained from the Indicated Single Cut to the Marked Graph (a)

written as [GH85]:

$$\mathcal{X} = \mu(\frac{\rho - \rho^{K+1}}{1 - \rho^{K+1}}) \tag{4.1}$$

where $\rho = \lambda/\mu$ is the ratio between the rates λ and μ of the aggregated transitions. With the above presentation we devise a *divide and conquer* approach to compute an approximation of the throughput: *it should be obtained through an iterative scheme using both aggregated net systems and the basic skeleton system.* Assuming an initial guess for one of the aggregated services (e.g., transition τ_2), the throughput of $\mathcal{AS}_1$ (Figure 4.1.b) is computed (solving the underlying CTMC) and a value for the service τ_1 is determined such that the basic skeleton has the same throughput (using equation (4.1)). In this way we *approximate* the RT of SN_1. The value of the service time of τ_1, is then substituted into the other aggregated net system (Figure 4.1.c) and the throughput is computed (solving the underlying CTMC). A new value for the service time of τ_2 is determined such that the throughput of the basic skeleton and $\mathcal{AS}_2$ are equal. The schema is iterated until a reasonable convergence of the throughputs is obtained.

The goal of the above presentation is just to convey the basic idea. The next sections formalize the technical aspects of single cuts. First in Section 4.3 the fundamental single input/single output (SISO) cut is characterized. The other possible single cuts are represented in Section 4.4 (SIMO/MISO cut) and Section 4.5 (MIMO cut).

4.3 Single Input/Single Output (SISO) Cut

This section is devoted to the characterization and analysis of the case where a net is split into two subnets in such a way that the interface is defined by a subset of input or output places of a unique transition and by a subset of input or output places of another (eventually the same) transition.

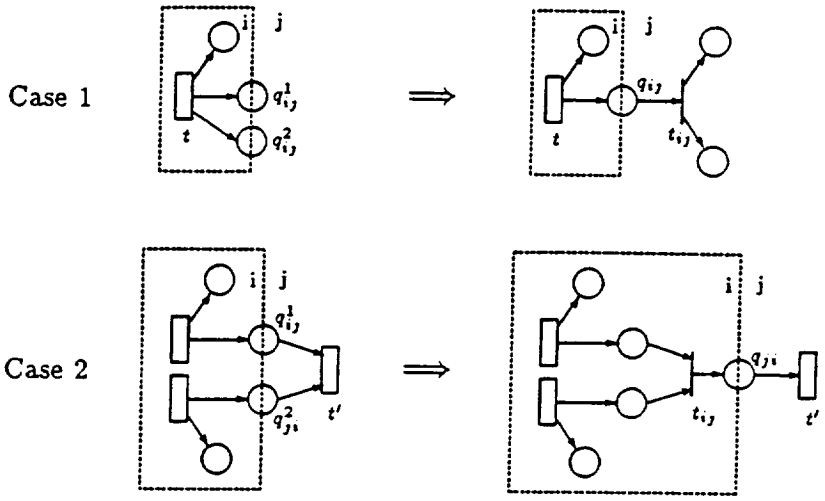


Figure 4.2: Canonical Transformations Presented on the Interface I_{ij} (from SN_i to SN_j: t_{ij} is an Immediate Transition)

4.3.1 SISO and Canonical SISO cuts

The idea of the single cut was formalized in Definition 3.3.1. As all places in MGs are persistent, it is possible to further characterize the interface:

Definition 4.3.1 (Interface) *Let ξ be a single cut traced through a set of interface places Q (Definition 3.3.1). $Q = Q_{12} \cup Q_{21}$ where Q_{ij} is the subset of places defining the interface from SN_i to SN_j, i.e., ${}^\bullet Q_{ij} \subseteq T_i$ and $Q_{ij}^\bullet \subseteq T_j$. t is a source (sink) transition of SN_i iff $t \in T_i$ and $t \in {}^\bullet Q_{ij}$ ($t \in Q_{ji}^\bullet$).*

Eventually source and sink transitions will be the same.

Definition 4.3.2 *Let ξ be a single cut producing subnets SN_1 and SN_2, connected through the subset of interface places Q_{12}, $Q_{21} \subseteq P$. Furthermore, let I_{ij} be the condition that the interface from SN_i to SN_j has to satisfy. ξ is SISO if:*

$$I_{12} : \exists t_{12} \in T_1 : Q_{12} \subseteq t_{12}^\bullet \text{ or } \exists t'_{12} \in T_2 : Q_{12} \subseteq {}^\bullet t'_{12}$$

$$I_{21} : \exists t_{21} \in T_2 : Q_{21} \subseteq t_{21}^\bullet \text{ or } \exists t'_{21} \in T_1 : Q_{21} \subseteq {}^\bullet t'_{21}$$

Both subconditions in I_{ij} are illustrated in Figure 4.2. Moreover Figure 4.2 illustrates a *canonical transformation* to deal with the case where we have multiple places interfacing to the same pre-set (post-set) of a transition (i.e., when $\mid Q_{12} \mid > 1$ or $\mid Q_{21} \mid > 1$). In the transformed net each interface has only a single place. We introduce an *immediate* transition t_{ij} and replace the interface Q_{ij} by a single place q_{ij}.

The canonical transformation preserves the underlying CTMC (only vanishing markings are introduced). Conceptually we could transform any SISO cut into a canonical SISO cut (i.e., through two different interface places, q_{12} and q_{21} which connect SN_1 to SN_2 and vice versa), but the canonical transformation increases the vanishing state space. Therefore, we would not introduce the immediate transition t_{ij} in Figure 4.2 for the generation of the CTMC. In order to simplify notation and statements, *in the rest of this section we assume without loss of generality, that the cut is canonical* (i.e., done through a single input place and a single output place) or that the canonical transformation is done whenever necessary (even if only conceptually, because of the increased vanishing state space). Therefore, in the rest of this section $\mid Q_{12} \mid = \mid Q_{21} \mid = 1$, $Q_{12} = \{q_{12}\}$ and $Q_{21} = \{q_{21}\}$.

Property 4.3.1 (SISO cut)

i) It is possible to check whether a cut is SISO is in linear time.

ii) The canonical transformation preserves the CTMC associated to the original system.

4.3.2 Qualitative Aspects

The process of obtaining aggregated net systems $\mathcal{AS}$ (Figure 4.1), can be viewed as a *reduction of the net system* [Ber85, Sil85]. Ideally speaking, we look for a reduction rule preserving the performance indices, in particular the throughput.

Rules preserving the throughput will be very limited. If the throughput cannot be preserved, what property should be reasonable to preserve in such a way that our iterative scheme yields a "good" approximation for the throughput? We decide to preserve the *maximum number of tokens in the non-aggregated places* (i.e., its marking bound). This corresponds to the idea of preserving the number of customers of the original system to its abstract views: the aggregated nets and the basic skeleton system. The preservation of the number of customers will imply the preservation of liveness for MG systems (remember that non-liveness is equal to null throughput!).

The work will proceed in two steps: First the *qualitative* aspects will be stated in this section using known facts from the structural theory of MG systems, and later the *quantitative* aspects will be addressed (Section 4.3.3).

Once a SISO cut has been decided on by the analyst, the transformation of the original net system into an aggregated net system can be viewed as a reduction process in which one subnet subsystem is aggregated into a single transition preserving the interface places and the other subnet subsystem. The marking of the remaining places has to be computed.

Let $\mathcal{N} = (P, T, F)$ be an MG and ξ a canonical SISO cut through places q_{12} and q_{21} defining two subnets $SN_i = (P_i, T_i, F_i)$ with $i = 1, 2$, where $Q = \{q_{12}, q_{21}\}$.

The original net system $\langle \mathcal{N}, M_0 \rangle$ leads to two aggregated net systems, $\mathcal{AS}_i = \langle \mathcal{AN}_i, M_0^i \rangle$ $i = 1, 2$, and the basic skeleton system. Structurally speaking from $\mathcal{N}$, SN_j is reduced to τ_j in $\mathcal{AN}_i$ (here $\mathcal{AN}$ refers to the structure, while $\mathcal{AS}$ refers to $\mathcal{AN}$ plus some initial marking). The decomposition process is formalized by the following rule.

Rule 1: SISO Decomposition Rule

1. **Structure:** $\mathcal{AN}_i = (\tilde{P}_i, \tilde{T}_i, \tilde{F}_i)$, $i = 1, 2$ is an aggregated net, where:
   - $\tilde{P}_i = P_i$

- $\tilde{T}_i = T_i \cup \{\tau_j\}$
- $\tilde{F}_i = F_i \cup (q_{ij} \times \tau_j) \cup (\tau_j \times q_{ji})$; $(\Rightarrow q_{ij} = {}^\bullet\tau_j, q_{ji} = \tau_j^\bullet)$

The basic skeleton is just the cycle $\{q_{12} \rightarrow \tau_2 \rightarrow q_{21} \rightarrow \tau_1 \rightarrow q_{12}\}$

2. **Marking:**

Let $\overline{M}$ be an optimal solution to the following linear programming problem:

$$\begin{aligned} &\textit{maximize} && \overline{M}(q_{21}) \\ &\textit{subject to} && \overline{M} = M_0 + C \cdot \vec{\sigma} \geq 0 \\ & && \vec{\sigma} \geq 0, \ \overline{M} \geq 0 \end{aligned} \qquad \text{(LPP3)}$$

where C is the incidence matrix of the canonical MG obtained from the original net and cut. The initial markings for the two aggregated nets and basic skeleton are the projection of $\overline{M}$:

- for $\mathcal{AN}_i$: $\widetilde{M}_0^i(p) = \overline{M}(p), \forall p \in \tilde{P}_i$
- for the basic skeleton: $M_0^s(q_{12}) = 0$ and $K = M_0^s(q_{21}) = \overline{M}(q_{21})$

Property 4.3.2 (SISO Reduction Rule) *Let $\langle \mathcal{N}, \mathcal{M}_0 \rangle$ be a live MG.*

1. *For any reduction (from the original net system to an aggregated net system or to the basic skeleton system) the bounds of the places are preserved: $B(p) = \tilde{B}(p), \forall p \in P_i$*
2. *Liveness in both aggregated net systems and the basic skeleton system is preserved.*

Proof: It is a direct consequence of the reachability and liveness theorems for strongly connected MGs (see Section 3.4.2). The transformation of the marking empties the least-loaded path in the subnet which is being reduced, therefore we

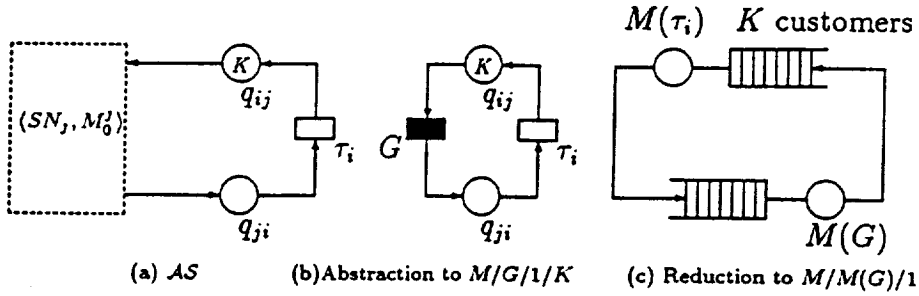


Figure 4.3: Abstraction of $\mathcal{AS}$ and Throughput Preserving Reduction

maintain the marking bound of places. Thus, the liveness of the system is preserved. ∎

Moreover, it is not difficult to realize that *for any optimal solution of (LPP3)* $\overline{M}(q_{12}) = 0$. This is true because q_{12} and q_{21} belong to the same set of cycles (P-semiflows). Thus, the maximization of $\overline{M}(p_{ij})$ implies the minimization of $\overline{M}(p_{ji})$.

Because LPPs are of polynomial time complexity, polynomial time algorithms lead to the aggregated net systems and to the basic skeleton systems.

4.3.3 Quantitative Aspects

During the RTA aggregation, we replace a subnet SN_i by an equivalent station (transition τ_i) with single server semantics. The response time of τ_i in isolation under an assumed arrival process approximates the response time of the replaced subsystem (i.e., subnet plus marking). In the aggregated net system $\mathcal{AS}_j$, the aggregation of subnet SN_i (represented by τ_i) is the generator of the arrival of tokens to the subnet SN_j, $i \neq j$. The service time associated with the transitions τ_i, $s(\tau_i) = 1/\mu_i$, $i = 1, 2$ represents an approximation of the mean completion time (MCT) of SN_i in the original MG system.

An interpretation of the system $\mathcal{AS}_j$ (Figure 4.3.a) is the classical $M/G/1/K$ queue, shown in Figure 4.3.b. Here we know the mean service time of the exponential server (transition τ_i), the number of customers in the system $K = M(q_{12}) + M(q_{21}) =$

$M(q_{21})$, and the throughput $\mathcal{X}$ of the system. Then our goal is to find an equivalent server for G (where G is a general distribution function which summarizes the behavior of the subsystem $\langle SN_j, M_0^j \rangle$).

One possible solution is to approximate $M/G/1/K$ by an $M/M(G)/1/K$ queue (Figure 4.3.c), *preserving the throughput* of the original system (here $M(G)$ expresses the fact, that the general server G is approximated by the exponential server $M(G)$). Then the problem at hand is the following: with the information from the analysis of the $M/G/1/K$ queue (population K, interarrival rate λ and throughput $\mathcal{X}$), we have to find the service rate μ of $M(G)$ such that the throughput of $M/G/1/K$ and $M/M(G)/1/K$ is the same. Equation (4.1) can be rewritten as:

$$\mathcal{X} = \lambda(1 - \frac{1}{\sum_{i=0}^{K} \rho^{-i}}) \tag{4.2}$$

It is easy to see that $\mathcal{X} < \lambda$, because the throughput cannot be higher than the fastest firing rate of a transition. Rewriting equation (4.2) as polynomial and defining $\beta = \mathcal{X}/\lambda$, $\beta < 1$, we have to find the positive solution of the following polynomial:

$$f(\frac{1}{\rho}) = \sum_{i=1}^{K} \frac{1}{\rho^i} - \frac{\beta}{1-\beta} = 0 \tag{4.3}$$

The polynomial $f(\frac{1}{\rho})$ has a unique positive real solution because:

$$f(0) = -\frac{\beta}{1-\beta} < 0 \tag{4.4}$$

$$\frac{\mathrm{d}f(\frac{1}{\rho})}{\mathrm{d}(\frac{1}{\rho})} = \sum_{i=1}^{K} i(\frac{1}{\rho})^{i-1} > 0, \ \forall \frac{1}{\rho} > 0 \tag{4.5}$$

i.e., the function is strictly increasing in $1/\rho$. The unknown firing rate is obtained by setting $\mu = \lambda/\rho$.

With this mapping of a $M/G/1/K$ to a $M/M(G)/1/K$ queue, we maintain the response time of the system when $K = 1$ because the steady-state probabilities of an $M/G/c/c$ and an $M/M/c/c$ queue are the same [GH85]. Therefore, the throughput and the response time of both queues are the same if $K = 1$. When $K > 1$, we only

approximate the RT. Note that in the original work of RTP, the isolated subsystem was analyzed under an open arrival process, while here we analyze the subsystem as a *closed* system.

We can define the following functions:

function *solve_basic_skeleton*($\lambda, \mathcal{X}$)

{Solves basic skeleton with unknown rate μ}

input: λ: known arrival rate

$\mathcal{X}$: throughput

returns: μ

function *solve_aggregated_system*(μ, k)

{Solves the underlying CTMC of $\mathcal{AS}_k$}

input: k: indicates system to be solved

μ: firing rate of aggregated transition τ

returns: $\mathcal{X}_k$

function *test_convergence*($\mathcal{X}_{new}$, $\mathcal{X}_{old}$)

{checks whether a convergence criterion has been achieved}

input: $\mathcal{X}_{new}$: most recently computed throughput

$\mathcal{X}_{old}$: previously computed throughput

returns: TRUE if convergence has been achieved

FALSE otherwise

Algorithm 1: Quantitative analysis of SISO cuts

1) Select a cut ξ

2) Generate the aggregated net systems $\mathcal{AS}_1$ and $\mathcal{AS}_2$ and the basic skeleton system using the Decomposition Rule.

3) Initial Value computation: Determine an initial value of the firing rate of τ_1, μ_1^0 (belonging to $\mathcal{AS}_2$). A reasonable initial value can be obtained as follows:

Let τ_2 be an immediate transition (i.e., $\mu_2 = \infty$) in $\mathcal{AS}_1$.

$\mu_1^0 := 1/\Gamma_1$;

where Γ_1 is given by (LPP2) (Theorem 3.4.4).

$\mathcal{X}_2^0 :=$ *solve_aggregated_system*$(\mu_1^0,\ 2)$;

4) $r := 0$; {iteration steps}

5) repeat

$r := r + 1$;

$\mu_2^{r-1} :=$ *solve_basic_skeleton*$(\mu_1^{r-1}, \mathcal{X}_2^{r-1})$;

$\mathcal{X}_1^r :=$ *solve_aggregated_system*$(\mu_2^{r-1},\ 1)$;

convergence := *test_convergence*$(\mathcal{X}_1^r,\ \mathcal{X}_2^{r-1})$;

if *convergence* = FALSE then

$\mu_1^r :=$ *solve_basic_skeleton*$(\mu_2^{r-1}, \mathcal{X}_1^r)$;

$\mathcal{X}_2^r :=$ *solve_aggregated_system*$(\mu_1^r,\ 2)$;

convergence := *test_convergence*$(\mathcal{X}_2^r,\ \mathcal{X}_1^r)$;

end if

until *convergence* = TRUE

Algorithm 1 implements an *iterative* method. Thus, the *existence* and *uniqueness* of the solution, and the *convergence* of the method should be addressed. Extensive tests have shown that for all cases considered, there exists one and only one solution, which is typically computed between 2 and 6 steps, until the difference between the two last estimations of the throughput is less than 0.1% (the convergence criterion is computed in the function *test_convergence*).

Although experimental convergence does not seem to depend on the initial

values for the iteration, "good initial values" can reduce the number of iterations, thus the practical complexity. The throughput of deterministic timings is frequently relatively close to the case of exponential timings, and its computation is polynomial (using Theorem 3.4.4). Therefore, the choice of the initial value of τ_1 is justified. Because of shorting out and the assumption of deterministic time, the initial value is an optimistic initial value. From our experience, the cycle time of the subnet in short circuit is in many practical cases within 20% of the actual value.

4.4 Single Input/Multiple Output (SIMO) Cut

This section considers a class of cuts (disjoint with respect to the SISO class) where the places of *one and only one* of the two interfaces does not belong both to the pre-set or the post-set of a single transition.

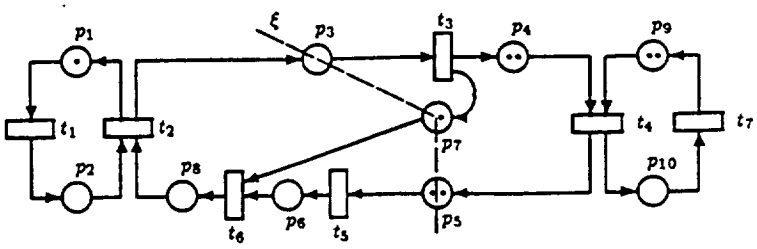
The output interface of SN_2 in Figure 4.4.a, $\{p_5, p_7\}$, is a multiple interface. More precisely stated, the pre- and the post-conditions of the interface Q_{21} have two transitions: $\{t_3, t_4\}$ and $\{t_5, t_6\}$, respectively. The input interface leading to SN_2 is defined through a single place p_3. Thus, the cut is SIMO with respect to SN_2. On the other hand, if SN_1 is taken as the reference subnet, the same cut is Multiple Input/Single Output (MISO, the symmetrical case). Therefore, in the rest of this work we refer to these types of cuts as *SIMO cuts.*

Definition 4.4.1 *Let ξ be a single cut producing subnets SN_1, and SN_2, ξ is SIMO with respect to SN_2 (MISO with respect to SN_1) if:*

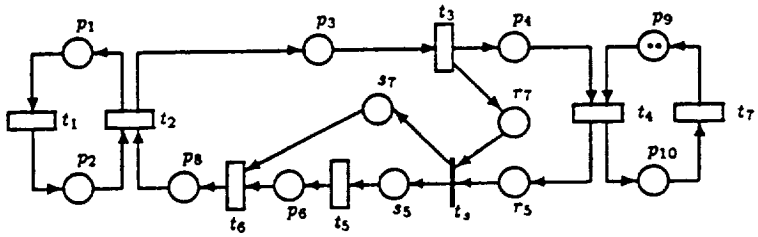
$$I_{12} : \exists t_{12} \in T_1 : Q_{12} \subseteq t_{12}^{\bullet} \text{ or } \exists t'_{12} \in T_2 : Q_{12} \subseteq^{\bullet} t'_{12} \quad \{\textit{single interface}\}$$

$$\overline{I}_{21} : \nexists t_{21} \in T_2 : Q_{21} \subseteq t_{21}^{\bullet} \text{ and } \nexists t'_{21} \in T_1 : Q_{21} \subseteq^{\bullet} t'_{21} \quad \{\textit{multiple interface}\}$$

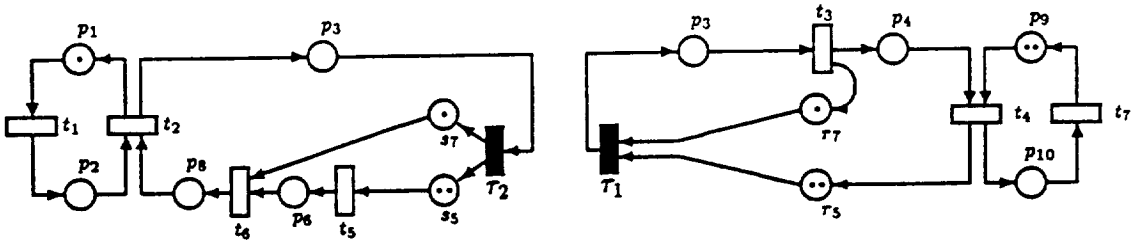
Comparing Definition 4.3.2 and 4.4.1, it is clear that a SIMO cut simply negates the second condition of the SISO cut. Thus, SISO and SIMO are disjoint classes of cuts.



(a) Original net system and cut



(b) Introducing the shadow transition t_s associated to the cut ξ



(c) The two aggregated net system $\langle \mathcal{AN}_i, M_0^i \rangle$.
Places r_k or s_k represent p_k of the original net

Figure 4.4: SIMO Cut and Aggregated Net Systems

We already know how to deal with the SISO case. Thus, an idea is to transform the underlying net structure in such a way that the resulting subnets have SISO characteristics. The necessary transformation on the net structure leads to the introduction of a single immediate *shadow transition* t_S.

Definition 4.4.2 (shadow transition) *Let Q_{ij} be the set of multiple interface places generated by a cut ξ. The duplicate interface places R_{ij}, S_{ij} and the shadow transition t_s are generated as follows:*

$$\forall q'_{ij} \in Q_{ij}, \quad {}^\bullet r'_{ij} = {}^\bullet q'_{ij}$$

$$s'^\bullet_{ij} = q'^\bullet_{ij}$$

$$r'^{\bullet}_{ij} = {}^{\bullet}s'_{ij} = t_s \tag{4.6}$$

Figure 4.4.b shows the introduction of a shadow transition t_S leads to the duplication of the interface $\{p_5, p_7\}$ creating the places $\{r_5, r_7\}$ and $\{s_5, s_7\}$. Then we have:

$${}^{\bullet}t_S = \{r_5, r_7\}, \quad t_S^{\bullet} = \{s_5, s_7\}$$

The cuts defined now through $q_{12} = \{p_3\}$ and $Q_{21} = \{r_5, r_7\}$ or $Q_{21} = \{s_5, s_7\}$ are SISO. From a structural point of view, the introduction of *a shadow transition* means the introduction of *a new synchronization constraint.* In general, the presence of the shadow transition creates new elementary cycles. In our case:

(α) $p_3 \rightarrow t_3 \rightarrow r_7 \rightarrow t_s \rightarrow s_5 \rightarrow t_5 \rightarrow p_6 \rightarrow t_6 \rightarrow p_8 \rightarrow t_2 \rightarrow p_3$

(β) $p_3 \rightarrow t_3 \rightarrow p_4 \rightarrow t_4 \rightarrow r_5 \rightarrow t_s \rightarrow s_7 \rightarrow t_6 \rightarrow p_8 \rightarrow t_2 \rightarrow p_3$

are new cycles. They lead to the following P-invariants:

(α) $M(p_3) + M(r_7) + M(s_5) + M(p_6) + M(p_8) + M(p_3) = K_\alpha(M_0)$

(β) $M(p_3) + M(p_4) + M(r_5) + M(s_7) + M(p_8) + M(p_3) = K_\beta(M_0)$

where $K_\alpha(M_0)$ and $K_\beta(M_0)$ is the number of tokens in the newly created P-invariants.

Using the standard reduction approach on the SISO cut in Fig 4.4.b, the aggregated nets in Fig. 4.4.c are obtained. Observe that both can easily be derived from the original net. In our presentation we introduce the transformed net with the shadow transition to highlight the presence of the additional synchronization constraint.

Now we have to choose appropriate initial markings for the aggregated nets (Fig. 4.4.c). To better understand the problems, let us go back to the original net system, and the transformed net with the shadow transition. One solution could be

derived by maximizing the number of tokens in the places of the multiple interface of the original net system (i.e., $\{p_5, p_7\}$). Such a solution is, for example: $M^T = (1,0,0,0,2,0,1,0,2,0)$. Observe that $M(p_5) = B(p_5) = 2$, $M(p_7) = B(p_7) = 1$ and $M(p_3) = 0$.

The following property formalizes the previous discussions:

Property 4.4.1 *Let $(\mathcal{N}, \mathcal{M}_0)$ be a live MG system, ξ a SIMO cut with the single interface q_{12} and the multiple interface Q_{21}. Any solution of the marking of the multiple interface places:*

$$\begin{array}{ll} max & \sum_{p \in Q_{21}} M(p) \\ subject\ to & M = M_0 + C \cdot \vec{\sigma} \geq 0 \\ & \vec{\sigma} \geq 0, \ \ M \geq 0 \end{array} \qquad \text{(LPP4)}$$

is such that $\forall p \in Q_{21}$, then $M(p) = B(p)$ and $M(q_{12}) = 0$.

With a solution of LPP4, the number of tokens in the places belonging to the multiple interface is their marking bound.

Proof: It is easy to see that any elementary circuit through q_{12} should go through one and only one place of the interface Q_{21}. Thus, reasoning on the P-invariants generated by these circuits and considering the reachability theorem (Theorem 3.4.3), it can be concluded that maximizing the marking on Q_{21} leads to $M(q_{12}) = 0$.

On the other hand it can be proved that there exists no elementary circuit containing two (or more) places of Q_{21}: from $p \in Q_{21}$ we should reach q_{12} before returning to any place of Q_{21}. Thus, a circuit through two places of Q_{21} should go twice through q_{12} and the circuit is not elementary.

The elementary P-invariants generated by the circuits containing places of Q_{21} have one and only one place of Q_{21} (i.e., they do not interfere). Thus, maximizing the sum of tokens is equivalent to maximizing the number of tokens in each place

of Q_{21}. The result follows because of the reachability theorem (Theorem 3.4.3). ∎

Once a solution of (LPP4) has been computed, the problem of where the tokens should belong to once the structure of the net is transformed arises:

- **Case a:** tokens into ${}^\bullet t_S$, i.e., into the pre-set of the shadow transition, $M_0(r_5) = 2$, $M_0(r_7) = 1$, $M_0(s_5) = M_0(s_7) = 0$.

- **Case b:** tokens into $t_S^\bullet$, i.e., into the post-set of the shadow transition, $M_0(r_5) = M_0(r_7) = 0$, $M_0(s_5) = 2$, $M_0(s_7) = 1$.

In the original net system we have the following marking bounds:

$B(p_4) = B(p_5) = 2$; $B(p_3) = B(p_7) = B(p_8) = 1$.

After the computation of the new markings we have the following changes in the marking bounds:

Case(a), tokens in pre-set of t_S: $B(s_5)=1$, $s_5 \in SN_1$

Case(b), tokens in post-set of t_S: $B(p_4)=1$, $p_4 \in SN_2$.

Because of the shadow transition, in both cases the behavior of the resulting net system is different from the original. A closer examination shows that computing the marking according to (a), we only preserve the marking bound of the places in SN_2. On the other hand if we compute the marking according to (b) only the marking bounds of the places in SN_1 are preserved.

In case (a) ((b) respectively) the initial marking for the transformed net with the shadow transition preserves the bounds of the places belonging to SN_2 (SN_1) if tokens corresponding to the original interface places Q_{21} are deposited in the pre-set (post-set) of the shadow transition, ${}^\bullet t_s = \{r\}$ ($t_s^\bullet = \{s\}$).

For the computation of the two aggregated net systems and the basic skeleton system we do not need to introduce the shadow transition explicitly (after all it leads to an increase of the vanishing state space). The process leading to the aggregated

net systems and basic skeleton system can be expressed directly by means of the following decomposition rule. In this way, all the marking bounds are preserved. The effect of its application is shown in Fig. 4.4.c.

Rule 2: Multiple Interface Decomposition Rule

1. **Structure:**

   (a) $\mathcal{AN}_i = (\tilde{P}_i, \tilde{T}_i, \tilde{F}_i)$ $i = 1,2$ is an aggregated net, where:

   - $\tilde{P}_i = P_i$
   - $\tilde{T}_i = T_i \cup \{\tau_j\}$
   - $\tilde{F}_i = F_i \cup (Q_{ij} \times \tau_j) \cup (\tau_j \times Q_{ji})$, $(\Rightarrow \quad {}^\bullet\tau_j = Q_{ij}, \quad \tau_j{}^\bullet = Q_{ji})$

   (b) The basic skeleton is just the cycle $\{q_{12} \rightarrow \tau_2 \rightarrow q_{21} \rightarrow \tau_1 \rightarrow q_{12}\}$, where q_{ij} now represents the subset of places Q_{ij}.

2. **Marking:**

   Let $\overline{M}$ be a solution of the following linear programming problem on the original net system (a generalization of (LPP3)):

$$\begin{aligned} \overline{M} = \quad & \text{maximize} \quad E_{21} \cdot \overline{M} \\ & \text{subject to} \quad \overline{M} = M_0 + C \cdot \vec{\sigma} \geq 0 \\ & \qquad\qquad\quad \vec{\sigma} \geq 0 \end{aligned} \tag{LPP5}$$

   where E_{21} is the characteristic vector of the multiple interface, Q_{21}, i.e.:

$$E_{21}(p) = \begin{cases} 1 & \text{if } p \in Q_{21} \\ 0 & \text{otherwise} \end{cases}$$

   The initial marking of $\mathcal{AN}_i$, $i = 1,2$, is $\widetilde{M}_0^i$, the projection of $\overline{M}$ on $\mathcal{AN}_i$, i.e., $M_0^i(p) = \overline{M}(p)$ $\forall p \in P_i$, considering that places r_k and s_k represent p_k of the original net. The initial marking for the basic skeleton is:

$$M_0(q_{12}) = 0 \text{ and } M_0(q_{21}) = \min_{p \in Q_{21}} \{\overline{M}(p)\} = B(q_{21})$$

The multiple interface reduction rule is a generalization of the SISO rule (Rule 2). For SISO interfaces both rules lead to the same transformation. We can identify the projection of the marking vector on $\mathcal{AN}_1$ as *case (a)* (tokens in ${}^\bullet t_S$), while the projection on $\mathcal{AN}_1$ corresponds to *case (b)* (tokens in $t_S^\bullet$).

Property 4.4.2 *Let $\langle \mathcal{N}, \mathcal{M}_0 \rangle$ be a live MG system, ξ a SIMO cut on $\mathcal{N}$, and $\langle \mathcal{AN}_i, \widetilde{\mathcal{M}}_0^i \rangle$, $i = 1, 2$, the partially aggregated net systems obtained by applying the Multiple Interface Decomposition Rule, then:*

1. $B(p) = \tilde{B}_i(p)\ \forall p \in P_i$.
2. $\langle \mathcal{AN}_i, \widetilde{\mathcal{M}}_0^i \rangle$, $i = 1, 2$ *is live.*

Proof:

1. Any circuit of $\mathcal{AN}_i$ is the projection of a circuit of the original net (observe that all the new circuits introduced by the shadow transitions disappear with the subsequent reduction processes). The result follows because of the reachability theorem and the P-invariants induced by the circuits.

2. The bounds being preserved, all the circuits of the aggregated nets should be marked. Therefore, the aggregated net systems are live. ∎

In order to approximate the throughput, the initial value computation and iterative schema of the SISO case (see Algorithm 1, steps 3 and 4) can now be used with the two aggregated net systems and the basic skeleton system. Now the number of customers $K = B(q_{12})$, i.e., K is given by the marking bound of the single interface place, q_{12}.

It can easily be verified that the two aggregated net systems obtained with the Multiple Interface Decomposition Rule can be derived from the same transformed net (with the shadow transition, Figure 4.4.b) but for two different initial markings (i.e., two different net systems).

4.4.1 Alternative Interpretation of the SIMO cut

Another way of interpreting the SIMO decomposition is as follows: Is it possible to reconstruct the original net system from the two aggregated net systems $\mathcal{AS}_1$ and $\mathcal{AS}_2$ obtained with the Multiple Interface Decomposition Rule? As we will we see not, because we have made a structural change by introducing the shadow transition. It is possible to construct a net system where a *SISO* cut leads to the same aggregated net systems as with the SIMO cut. Let the initial marking of the net shown in Figure 4.4.b be $M(s_7) = M(r_5) = M(p_1) = 1$ and $M(s_5) = M(p_9) = 2$. The SISO cut through the places s_7, s_5 and p_3 together with the canonical transformation also leads to to the aggregated net systems shown in Figure 4.4.c but its behavior is completely different from that in Figure 4.4.a. Here we have introduced the shadow transition and *added* tokens until the marking bound of all places is the same as in the corresponding places of Figure 4.4.a.

Theorem 4.4.1 *Let $\langle \mathcal{N}, M_0 \rangle$ be a MG system with the SIMO cut ξ defined through the single interface q_{ij} and the multiple interface Q_{ji} and $\mathcal{N}'$ the net structure after introducing the shadow transition.* The SISO cut ξ' through $\langle \mathcal{N}', M_0' \rangle$ with the interface places R_{ji} or S_{ji} produces the same aggregated net systems as the cut ξ in $\langle \mathcal{N}, M_0 \rangle$. *M_0' is given as follows: Let $\overline{M}(p)$ be a solution of LPP5. M_0' is the projection of $\overline{M}(p)$ on P' considering that R_{ji} now represents Q_{ji}. Furthermore, let k be the minimum number of tokens in the interface R_{ji}, i.e.,*

$$k = \min_{p \in R_{ji}} M(p) \tag{4.7}$$

The marking of S_{ji} is given by

$$\forall s'_{ji} \in S_{ji}, \quad M(s'_{ji}) = M(r'_{ji}) - k \tag{4.8}$$

Proof: We have to show that the marking bound of the places in $\langle \mathcal{N}', M_0' \rangle$ is the same as in $\langle \mathcal{N}, M_0 \rangle$.

1. The bound of places in SN_j cannot have *decreased* because we have at least maintained the number of tokens in each directed circuit by using the projection of the marking vector of LPP5 (R_{ji} is now part of SN_j).

2. The maximum number of tokens in a directed circuit in SN_j is the same because there exists at least one path in SN_i (between q_{ij} and S_{ji}) which is empty. By using corollary 3.4.4 we have not *increased* the marking bound of the places in SN_j.

The same reasoning holds for SN_i, because by firing t_s the situation of SN_j and SN_i is now symmetrical and the same argument holds. ∎

The above reasoning is conceptual, because the aggregated net systems can be obtained directly from the original net system by elementary net reduction rules, specifically the macrotransition reduction rule (the reverse-dual of the macroplace reduction rule presented in [Sil85]). Nevertheless, when interpreting the results generated by a SIMO cut, it is important to remember that a structural transformation has taken place.

Because of the shadow synchronization constraint (a pessimistic transformation) and the change in marking by Theorem 4.4.1 (an optimistic transformation), the global estimated performance may be pessimistic or optimistic.

4.5 Multiple Input/Multiple Output (MIMO) Cut

The third (and last) class of single cuts is obtained when the places of the input *and* the output interface do not belong to the pre-set or the post-set of a single transition. In other words, several (multiple) transitions define both interfaces.

Definition 4.5.1 *Let ξ be a single cut producing subnets SN_1 and SN_2, connected through the subsets of interface places $Q_{12}, Q_{21} \subseteq P$. ξ is MIMO if:*

$$\overline{I}_{12} \; : \; \nexists t_{12} \in T_1 : Q_{12} \subseteq t_{12}^{\bullet} \; \textit{and} \; \nexists t'_{12} \in T_2 : Q_{12} \subseteq {}^{\bullet}t'_{12}$$

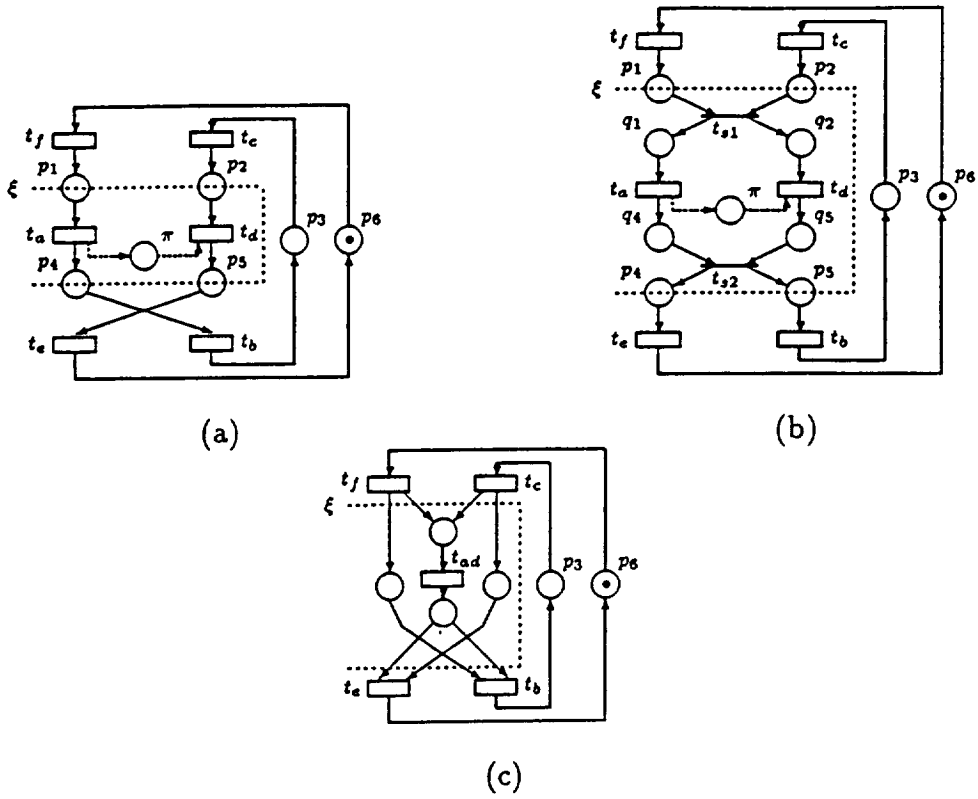


Figure 4.5: A Bad MIMO Cut and Two Possible Interpretations of the Net System Transformation

$$\overline{T}_{21} \; : \; \nexists t_{21} \in T_2 : Q_{21} \subseteq t_{21}^{\bullet} \; \textit{and} \; \nexists t'_{21} \in T_1 : Q_{21} \subseteq^{\bullet} t'_{21}$$

Comparing Definitions 4.3.2 and 4.5.1, it is clear that a MIMO cut negates both conditions of a SISO cut. SISO, SIMO and MIMO form a *complete* set of the disjoint classes of single cuts.

MIMO cuts can be much more problematic from a performance analysis point of view. In general, if we apply the technique presented previously, we do not maintain the meaning of the original net system. We will illustrate the previous statement with two examples.

The net system shown in Figure 4.5.a (without place π) is a monomarked (i.e., marked with a single token) elementary circuit. The MIMO cut presented is bad from any point of view: by making a structural transformation analogous to that introduced in Section 4.4 (i.e., introducing two shadow transitions, one per

interface), the net in Figure 4.5.b is obtained. New circuits have been introduced into the net system, which is now obviously dead. Adding a token to p_3, the net system becomes effectively live, but the behavior of the transformed system has nothing to do with that of the original system (e.g., compare the possible firing sequences). Another interpretation for the reduction process is illustrated in Figure 4.5.c where the behavior (taken as the firing sequence) is preserved, the only difference being that instead of firing a or d, an aggregated transition ad is fired. There are two problems associated with the second interpretation of the aggregation process:

(1) The aggregated net depends on the behavior of the original system and

(2) The aggregated net is not an MG.

What is the problem with the MIMO cut in Figure 4.5.a? The answer is quite simple: an elementary circuit has been cut in more than two places (four to be more precise, because the number of cuts on a circuit to split a net into two parts must be even). Doing that, two subcircuits of the same circuit are forced to be aggregated, something that is functionally meaningless.

In another way the MIMO cut of Figure 4.5.a is also problematic: we are "aggregating" two disconnected subnets p_1, t_a, p_4 and p_2, t_d, p_5 and from a performance point of view this may be dubious. Nevertheless, we can add an implicit place π to the original net (${}^\bullet\pi = t_a$ and $\pi^\bullet = t_d$). Because π is implicit [CS90], this new place does not change the behavior of the original system, and the subnet to be aggregated is now connected. But if we reduce the subnet we will change the parallel behavior ($t_a \parallel t_d$) in Figure 4.5.b to a sequential behavior ($t_a \circ t_d$). Therefore, two net systems (with and without the implicit place π) have identical behaviors, but lead to two transformed net systems with quite different behaviors! We can alleviate the problem by introducing the notion of *well formed cuts.*

Definition 4.5.2 (Well Formed Cut) *A MIMO cut ξ is said to be* Well Formed *if no elementary circuit is cut more than twice.*

Given a cut on an MG, it can be easily characterized as being well (or ill) formed. Let G_ξ be the arc weighted graph obtained from the MG considering the transitions as nodes and the places as arcs. Arcs are labeled with weight 1 if the corresponding place belongs to the interfaces defined by the cut, and 0 otherwise. Thus, the following can be concluded:

Property 4.5.1 *The MIMO cut ξ is well formed iff the maximum weighted circuit of G_ξ has weight two (i.e.,no circuit is cut more than twice).*

The problem of computing a maximum weighted circuit [Deo74] has polynomial time complexity, therefore, well formed cuts are characterized in polynomial time.

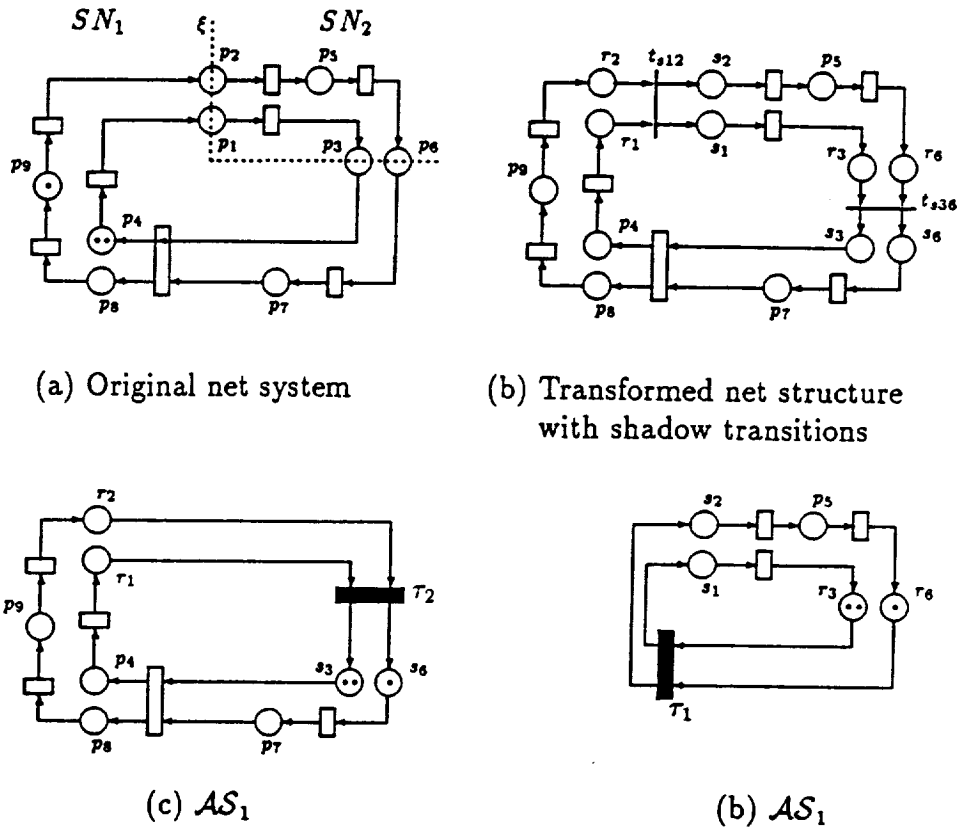
The transformation of the MIMO cut into a SISO cut leads to the introduction of *two* shadow synchronizations: one for the input and another for the output interface. Figure 4.5.b shows the structural transformation. Of course the shadow transitions can introduce many new circuits.

The process leading to two aggregated net systems and the basic skeleton system can be directly expressed by Rule 2, *Multiple Interface Decomposition Rule.* The two aggregated net systems produced by ξ in Figure 4.6.a are represented in Figure 4.6.c. The following are two interesting observations on our example. In the transformed net with the shadow transitions:

- Places p_4 and r_1 (representing p_1) in $\mathcal{AS}_1$ are now 1-bounded, while they were 2-bounded on the original net system, i.e., now the "number of customers" is not preserved.

- Place s_3 is implicit [CS90] and the behavior is not changed if we delete it.

The definition of the multiple interface decomposition rule emphasizes maximizing the number of tokens in the multiple interface. As the cut is now MIMO,



(a) Original net system

(b) Transformed net structure with shadow transitions

(c) $\mathcal{AS}_1$

(b) $\mathcal{AS}_1$

Figure 4.6: Example for MIMO Cut: Marking Bound Changes

both interfaces are multiple and a solution for $\overline{M}$ can be obtained by maximizing the number of tokens in Q_{21} or in Q_{12}.

The situation is now somewhat more symmetrical, what happens if we maximize the marking of the other multiple interface, i.e., $E_{12} \cdot \overline{M} = \sum_{p \in Q_{12}} M(p)$?.

The solution for $\overline{M}$ is now $\overline{M} = M_0$, thus:

- $M(r_1) = 2,\ M(r_2) = 1,\ M(s_3) = M(s_6) = M(p_4) = M(p_7) = M(p_8) = M(p_9) = 0.$
- $M(s_1) = 2\ M(s_2) = 1,\ M(r_3) = M(r_6) = M(p_5) = 0.$

The new initial marking for $\mathcal{AV}_2$ is reachable in $\mathcal{AS}_2$, therefore, basically the situation for $\mathcal{AS}_2$ has not changed. Nevertheless, $\mathcal{AS}_1$ has an initial marking

such that the bound of all places except s_3 is preserved! Moreover, even if $B(s_3)$ is different from $B(p_3)$ it is easy to see that s_3 is implicit [CS90] and can be removed without affecting the behavior! In other words, optimizing the marking on Q_{12} or on Q_{21} has different effects on the behavior of the aggregated net systems!

Property 4.5.2 *Let $\langle \mathcal{N}, \mathcal{M}_0 \rangle$ be a live MG system, ξ a* well formed *MIMO cut on $\mathcal{N}$, and $\langle \widetilde{\mathcal{N}}, \widetilde{\mathcal{M}}_0 \rangle$ the net system obtained applying the multiple interface decomposition rule. Then $\langle \widetilde{\mathcal{N}}, \widetilde{\mathcal{M}}_0 \rangle$ is live, but in general the bounds of places are not preserved.*

Proof: By maximizing the number of tokens in the output interface, we guarantee that each circuit is marked (the new circuits as well as the old ones, because each new elementary circuit has to include one input and one output interface place), therefore, the transformed net system is live. ∎

Anyhow, approximate quantitative results are obtained by applying the initial value computation and iteration schema of Algorithm 1 to the aggregated net systems and the basic skeleton system obtained with the *multiple interface decomposition rule* and maximizing the marking on Q_{21} or on Q_{12}. K is now the minimum number of tokens in the circuits which have been cut.

From the above discussion the following property follows:

Property 4.5.3 *The marking bound of $\mathcal{AS}_i$, $i = 1, 2$ is preserved, if the number of tokens in each cut cycle is the same.*

4.6 Multiple Cuts and Response Time Approximation: Hierarchical Decomposition

In Section 4.2 the fundamental ideas for an approximation technique based on single cuts have been introduced. Sometimes, the decomposition of the system

by a single cut into two subsystems is not sufficient, because we cannot solve the underlying CTMC's of the aggregated subsystems in a reasonable amount of time. Once again we have to look for other approaches to cope with the problem of state space explosion.

The basic idea is to consider not only a single cut, but to introduce several cuts. We consider that multiple cuts can be implemented in a *recursive* manner, leading to a hierarchy of subsystems which are structured in several levels.

In this approach we assume that the MG is decomposable into subnets by more than one single cut. In general, this assumption is not true, when we consider only a very restricted class of cuts like the SISO cut in this work. However, when we consider all possible kinds of single cuts (SISO, SIMO or MIMO) this decomposition is always possible.

Figure 4.7 illustrates the idea of the hierarchical decomposition. At each level the nets are recursively partitioned into two subnets by a single cut until the size of the state space of the resulting aggregated net systems is sufficiently small. Dashed transitions represent the aggregation upon which the service times are iterated at the current level. The approximate value of the throughput is computed in a *hierarchical manner.* The CTMCs associated with the aggregated net systems are computed at the lower level of the decomposition when they are small enough (systems $\mathcal{AS}^2_{11}, \mathcal{AS}^2_{12}, \mathcal{AS}^2_{21}$ and $\mathcal{AS}^2_{22}$ in Figure 4.7).

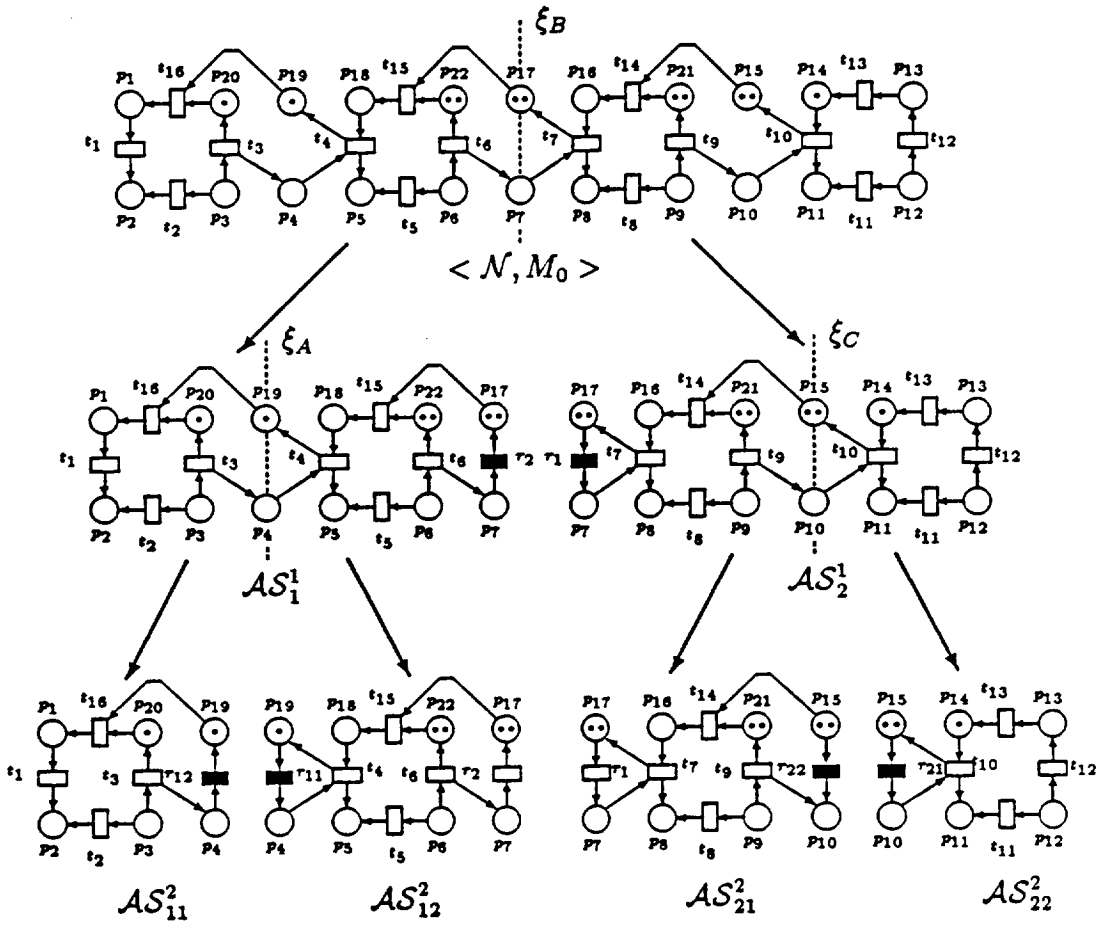


Figure 4.7: Hierarchical Decomposition into Various Levels

At each step of the recursive decomposition, the analyst must define a cut for each subnet, and evaluate the new set of aggregated subsystems whether it is possible to solve the associated CTMCs in a reasonable amount of time. Since an accurate estimation of the size of a CTMC is difficult (although a promising approach is outlined in [WD92a, WD92b]), the analyst should define when the CTMC is "small enough" to be solved analytically. Moreover, if an additional decomposition is needed, the analyst should define the new cut. Note that in this decomposition method the number of subsystems whose associated CTMC has to be solved is doubled with each decomposition.

We are now going to describe, in an intuitive way, the hierarchical decomposition method with the example presented in Figure 4.7. Because of the difficulties associated with MIMO cuts, in the following we only consider SISO and SIMO cuts.

Consider the original system denoted by $\mathcal{N}$ at the top of Figure 4.7. Then we choose a cut B through the places p_7 and p_{17}. This cut satisfies the SISO cut conditions. After the first cut is defined, we apply the decomposition process for a single cut presented in Section 4.3.2, and obtain the aggregated systems $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$.

Now we have to determine if the sizes of the state space of the aggregated systems $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$ are small enough. If the answer is affirmative, we can compute the throughput of the original system with the iterative process presented in Section 4.3.3. Note that if this condition holds at the first hierarchical level, we are applying the single cut method.

All systems that are not sufficiently "small" must be further decomposed. This means that for those systems we have to choose either a SISO or SIMO cut and apply the appropriate decomposition rule.

In the general case, the number of systems can be large and distributed in several levels. The number of systems and the number of levels are related. If we

have l systems, at most $l-1$ levels are implicated in the process. An abstract sketch of the recursive-interactive function for the quantitative analysis of the general case is presented below. All functions have been previously defined in Section 4.3.3.

Algorithm2 : Recursive – Interactive Analysis with SISO cuts

```
procedure solvesystem (net system, level)

inquire if the net system is "small enough";
if small enough then solve the underlying CTMC;
   else if   the net system has not been solved before
             then
                  request a new SISO cut;
                  use the decomposition rule to generate the left and right
                  aggregated net systems, and the basic skeleton system;
             endif
        repeat
             solvesystem(leftsystem, level+1);
             solve_basic_skeleton;
             solvesystem (rightsystem, level+1);
        until e_x ≤ ε(level)
end if
```

An important observation on the sketched algorithm is that the requirements of accuracy on the different levels of the recursion can be different. In the *test_convergence* function, we can use a convergence criterion dependent on the hierarchical level, i.e., ε(level). This is interesting in practice because lower levels are computed more frequently (they partially determine the time complexity) and the impact of less accurate results at lower levels is partially canceled at the upper

levels. The initial value problem for intermediate and lower levels is not detailed but the practice is to recover the last estimation at the corresponding level.

Property 4.6.1 (Multiple SISO Cuts) *Let $\langle \mathcal{N}, M_0 \rangle$ be a live MG system.*

1. *For every aggregated net system and for the basic skeleton system, the bounds of the places are preserved: $B(p) = \tilde{B}(p)$, $\forall p \in P_i$.*
2. *Liveness in all aggregated net systems and the basic skeleton system is preserved.*

Proof: Since the SISO decomposition rule is applied at each step of the hierarchical decomposition, Property 4.6.1 follows from Property 4.3.2. ∎

In the hierarchical decomposition, the full environment is recursively defined using the notion presented for the single cut. At each level, the environment of a subnet contains all information of the environment of previous levels in its branch. In other words, the error in the representation of the environment is bigger at deeper levels of the hierarchy. So we can say that the accuracy of the hierarchical method will be an inverse function of the number of levels in the hierarchical tree of decomposed systems.

Finally, recall that in the hierarchical decomposition, the approximated value of the throughput is computed by a recursive procedure. In order to solve each node of the decomposition tree, the children of the node must be solved. This means, that each iteration at level k implies a new computation of the throughput of all children at level k. So the analyst must evaluate the convenience of carrying out a new decomposition taking not only the size of the associated CTMC, but also the number of times the lower levels are evaluated into account.

4.7 Conclusions

In this chapter we have introduced the structural decomposition of MGs. It is based on a single cut through a single or multiple interface. We have seen that the

SIMO and MIMO cuts correspond to a *structural and marking transformation.* Furthermore, in the MIMO cut the marking bound of places is in general not preserved. This justifies that in the following we concentrate on SISO and to some extent on SIMO cuts. It is important to note that the basic skeleton is in *product form*, i.e., we have decomposed the MG system according to Definition 2.3.1 (product form decomposition). Based on the aggregated net systems and the basic skeleton, we can use both RTA, and any of the other techniques developed for "almost" PFQNs.

From the stochastic perspective we have introduced RTA, a new approximation technique for the case of single cuts. We have simplified RTP, so that in the proposed method the computation of the coefficient of variation is not required. RTA can be performed on a single level or can be used in conjunction with a hierarchical decomposition.

CHAPTER 5
Alternative Approximation Methods for Marked Graphs

5.1 Introduction

In the previous chapter we have explored the structural decomposition of MGs to derive the aggregated net systems. Based on the structural decomposition, we now present alternatives to RTA. In Chapter 4, we have presented Response Time Approximation (RTA), which is specifically designed for the single cut. In the following, we present Flow Equivalent Aggregation (FEA) and Marie's method. Both are well known approximation methods for QNs, which are here applied in the context of MG systems. They derive from the fact that by decomposing the MG system by a single cut, the basic skeleton is in product form.

Delay equivalence (DE) [LW91, WL91] is another approach for the iterative solution of MGs. Contrary to the previous approaches, it does not make use of the basic skeleton.

In this chapter, we present the approximation methods. In Chapter 6, all of the presented methods are compared with several examples.

5.2 Flow Equivalent Aggregation

In MGs, the basic approach is to substitute subnet SN_i by a flow equivalent transition (FET) τ_i, whose (state dependent) firing rate is characterized by the number of tokens in its input place q_{ji} (i.e., the queue length of the flow equivalent server). After the aggregation phase, both subnets are substituted by their respective FETs (τ_1 and τ_2). From a structural point of view, this leads to the basic skeleton. The overall performance analysis is done by solving the underlying CTMC.

5.2.1 Qualitative Aspects

The qualitative aspects of the decomposition have already been addressed in *Rule 1: SISO Decomposition Rule* (section 4.3.2) and *Rule 2: Multiple Interface Decomposition Rule* (section 4.4). Based on a single cut, we generate both aggregated nets. $\mathcal{AS}_1$ ($\mathcal{AS}_2$) will be used to analyze SN_1 (SN_2) under a varying load which allows us to formulate the flow equivalent transition τ_1 (τ_2).

5.2.2 Quantitative Aspects

In FEA, the subnets are analyzed in *short circuit*. This corresponds to setting τ_1 in $\mathcal{AS}_2$ (τ_2 in $\mathcal{AS}_1$) to an immediate transition. Alternatively we can *merge* the interface places and delete τ_i (this is preferred, because it reduces the number of vanishing markings). For the quantitative analysis, the number of tokens k in the interface place q_{21} has to be varied up to the maximum allowed, i.e., $k = 1 \ldots K$, where K is the initial marking of q_{21} and computed by solving LPP3.

Algorithm 2: Flow equivalent analysis of single cuts

1) Select a SISO cut ξ

2) Generate the aggregated net systems $\mathcal{AS}_1$ and $\mathcal{AS}_2$ in short circuit by merging the interface places as well as the basic skeleton system using the Decomposition Rule (SISO or Multiple Interface).

3) Analyze $\mathcal{AS}_i$, $i = 1,2$ with $m_0(q_{21}) = l$, $l = 1 \ldots K$ by solving the underlying CTMC and obtain the conditional throughputs $\mathcal{X}_1$ and $\mathcal{X}_2$.

4) Set the service rates of the aggregated transitions to the conditional throughput, i.e.,

$$\mu_i(l) = \mathcal{X}_i(l) \qquad i = 1,2; \;\; l = 1 \ldots K \tag{5.1}$$

5) Analyze the basic skeleton with the transition rates obtained from step 4.

The above algorithm yields the *maximum* reduction of the state space (higher than in RTA), but as we will see the accuracy can suffer. Alternatively, we can aggregate only one of the subnets (e.g., SN_1) and substitute the state dependent rate of μ_1 into $\mathcal{AS}_2$. The state space will be higher, but in general the quality of the approximation will be better.

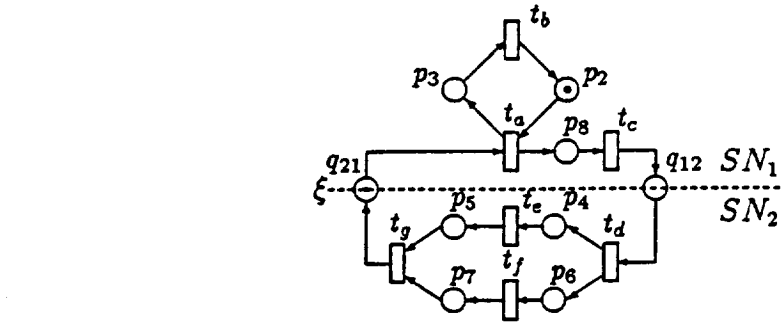
5.2.3 Flow Equivalent Aggregation vs. Response Time Preservation

In FEA the assumption is that *the service time depends only on the number of customers which are currently present in the subsystem.* In FEA the behavior of the subsystem is assumed to be *independent* of the arrival process and depends only on the number of customers in the system, i.e., the behavior is completely independent of the *environment.* This assumption is violated in several cases.

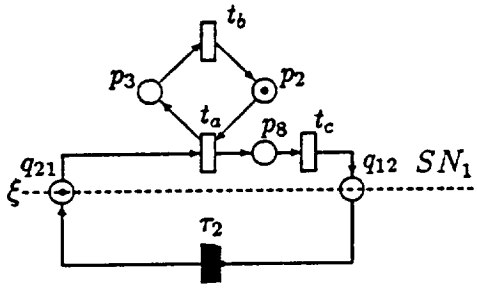
Let us illustrate the fact that even in very special cases the *mean completion* (or *traversing*) *time* (MCT) may depend on the interarrival process of the token. Informally, the MCT is the average time spent by a token traversing the subnet. Figures 5.1(b) and 5.1(c) show two 1-bounded MG systems. In the first (second) we assume τ_2 (τ_1) to have different rates. Figure 5.1(b) essentially represents, the circuit ($p_1 - t_a - p_2 - t_c - p_3 - \tau_2$) in which "the server" at t_a has a *setup time* modeled by t_b. Figure 5.1(c) is essentially a cyclic *fork-join* system.

The system in Figure 5.1(b) can be interpreted as a $M/G/1/1$ queue where τ_2 represents the exponential server, while SN_1, represents the general server. Note that the service time of G is just the MCT of SN_1 (a similar argument holds for the system in Figure 5.1(c)). The cycle time in an $M/G/1/1$ queue is simply the sum of the service times of M and G (*insensitivity*) [GH85]. The MCT of SN, Γ_{SN}, is therefore given by:

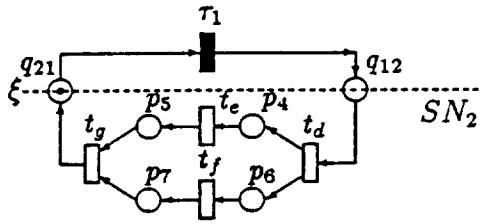
$$\Gamma_{SN} = \Gamma - \theta \tag{5.2}$$



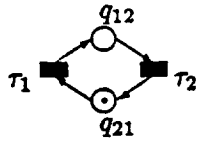
(a) $< \mathcal{N}, M_0 >$



(b) $\mathcal{AS}_1 = \langle \mathcal{AN}_1, \widetilde{M}_0^1 \rangle$



(c) $\mathcal{AS}_2 = \langle \mathcal{AN}_2, \widetilde{M}_0^2 \rangle$



(d) Basic Skeleton

Figure 5.1: Aggregated Net and Basic Skeleton Systems Obtained from the Cut ξ (*Service Times*: $s_a = 0.33$, $s_b = 1.0$, $s_c = 0.25$, $s_d = 0.20$, $s_e = 0.33$, $s_g = 0.5$)

where Γ is the mean interfiring (cycle) time with respect to any transition, and θ denotes the firing (service) time of any transition τ, $\theta = s_{\tau_1} = s_{\tau_2}$.

Table 5.1 presents the numerical data of the analysis of the systems in Figures 5.1(b) and 5.1(c). Both systems are 1-bounded, thus the variation in Γ_{SN_1} (from 1.383 to 0.583) can be explained only due to the *dependency* between the general (time dependent, in particular) and the exponential server. If θ is very large ($\theta \gg 10$), t_b will almost surely fire before the firing of τ_2 ends. Thus, $\Gamma = s_a + s_c + \theta$ and $\Gamma_{SN_1} = s_a + s_c$. If θ is zero (i.e., τ_2 is immediate), then Γ_{SN_1} will be much higher because now it depends on, s_b, the firing time of t_b, plus s_c. On the other

	System in Fig. 5.1(b)		System in Fig. 5.11(c)	
$\theta = s_{\tau_1} = s_{\tau_2}$	Γ^b	Γ_{SN_1}	Γ^c	Γ_{SN_2}
0	1.383	1.383	1.783	1.783
0.1	1.411	1.311	1.883	1.783
1.0	1.983	0.983	2.783	1.783
2.0	2.850	0.850	3.783	1.783
10.0	10.656	0.656	11.783	1.783
$\gg 10$	$\theta + 0.583$	$s_a + s_c = 0.583$	$\theta + 1.783$	1.783

Table 5.1: Dependence and Independence of the Mean Completion Time on the Interarrival Time

hand, the value of Γ_{SN_2} (Table 5.1) is constant, i.e., for the given initial marking there exists a perfect decoupling (independence) between the general (in this case time independent) and the exponential queues. This is easy to explain because the 1-bounded system is insensitive to the pdf of the server: both servers act as *delay* nodes.

The case of Figure 5.1(b) is just an illustration of the fact that *the existence of internal loops in a subnet may violate the independence condition.* Another situation where the independence condition can be violated is when *trapped tokens* exist in a fork- join. To illustrate this case, assume a token is added to p_4 in Figure 5.1(c). Even if p_4 and p_5 are now 2-bounded, q_{12} is 1-bounded and the general server "works" as with a single customer. Therefore, insensitivity in the $M/G/1/1$ queue remains and Eq(5.2) is valid. Nevertheless, with the modified marking the MCT of SN_2 is no longer independent of θ (see Table 5.2).

For the particular case of the examples (1-bounded exponentially timed MG systems), it is easy to observe a *monotonicity* property: *increasing the service time* θ (representing a complementary subsystem) *never leads to a bigger MCT for the subsystem under consideration* (see Tables 5.1 and 5.2 for Γ_{SN_1} and Γ_{SN_2}, respectively). This is easy to understand: the strong dependency between the general and the exponential servers is due to the fact that the general server (i.e., the one

$\theta = s_{\tau_1} = 1/\mu_1$	Γ^c	Γ_{SN_2}
0.25	1.556	1.306
0.5	1.757	1.257
1.0	2.198	1.198
2.0	3.141	1.141
4.0	5.097	1.097

Table 5.2: Γ_{SN_2} Depends on θ for the System in Figure 5.1(c) if a Token is Added to p_4

summarizing SN_i) is *time-dependent.* But as MGs are conflict-free, the increase of the interarrival time of tokens eventually allows the tokens in the subnet to progress towards the output places (i.e., q_{12} for SN_1 and q_{21} for SN_2), resulting in a smaller or equal (but never larger!) MCT.

Because of the insensitivity of $M/G/1/1$ queues, the estimated values of the cycle time Γ in FEA, (Γ_{FEA}) are as follows:

(a) Figure 5.1(b): $\Gamma^b_{FEA} = \theta + 1.383$ ($\neq \Gamma^b$, Tables 5.1)

(b) Figure 5.1(c): $\Gamma^c_{FEA} = \theta + 1.783$ ($= \Gamma^c$, Table 5.1, because of the independence)

If the FEA is performed on SN_1 for the marking in Figure 5.1(b), for $\theta = 2$, an error of 18.7% in the computation of the throughput is obtained.

In *response time preservation* (RTP) [ABS84] the response time of the subsystem is replicated. Here the assumption for the analysis phase is a Poisson arrival μ of customers representing the environment of the subsystem. During the aggregation phase the RT is replicated by an exponential server, i.e.,

$$RT_{SW}(\mu) = RT_{AN}(\mu) \tag{5.3}$$

The advantage of iterative methods lies in the fact that it can model the interaction of subsystems. The drawback is that convergence might eventually present a problem.

Figure 5.2: Open Subnetwork R^l_O

5.3 Marie's Method

The first step in Marie's method is the *Product Form Decomposition* of the Petri net according to Definition 2.3.1. For a SISO cut in a MG system, these conditions are obviously satisfied.

Let R^l be the l-th subsystem with n^l customers present, $\lambda^l(n^l)$ the state dependent arrival rate and K the total number of customers. The *open* system in isolation is denoted by R^l_O as shown in figure 5.2 and is analyzed in isolation under a state dependent Poisson arrival process.

Suppose that $\lambda^l(n^l)$ is known, then R^l_O can be analyzed. Note that although the system is nominally open, the arrival rate to the system is zero, once K customers are in the system. For the case of a single cut in MGs, the closed system has the structure of an aggregated net system. Now the transition τ_j (with firing rate $\lambda^l(n^l)$) is the generator of the *arrival* of customers to SN_i, $i \neq j$.

Particularly, we are interested in the steady state probability $Pr(n^l)$ of having n^l customers in the open subnetwork. Then the *conditional throughput* $\nu^l_O(n^l)$ can be obtained as [Mar79].

$$\nu^l_M(n^l) = \lambda^l(n^l - 1)\frac{Pr(n^l - 1)}{Pr(n^l)} \text{ for } n^l = 1, \ldots, K \tag{5.4}$$

As in FEA, the service rates $\mu^l(n^l)$ are set to the conditional throughput.

$$\mu^l(n^l) = \nu^l_O(n^l) \tag{5.5}$$

The problem is now reduced to finding the *state dependent arrival rates* $\lambda^l(n^l)$ for the open subnetwork R^l_O. To this end, the partitioned system (now in product

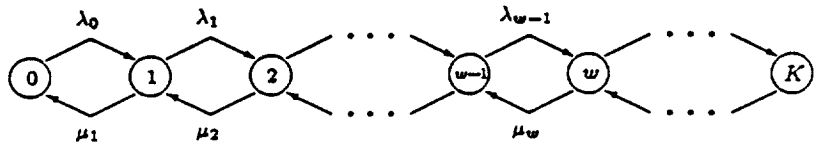


Figure 5.3: Birth-Death Process

form) is analyzed with the rates obtained from equation (5.5) (either by using mean value analysis [RL80] or simply the analysis of the CTMC).

$$\lambda^l(n^l) = \mu^l(n^l+1)\frac{P(n^l+1)}{P(n^l)} \quad n^l = 0,\ldots,K-1 \tag{5.6}$$

Having computed the state dependent arrival rates the iteration can be repeated.

An alternative interpretation of Marie's method for single cuts in MGs is the following: We analyze $\mathcal{AS}_i$ with a state dependent arrival rate $\lambda(n)$, $n = 1 \ldots K$ and obtain the conditional probabilities of having n customers in the subsystem. We then interpret $\mathcal{AS}_i$ as a general Markovian birth-death process as illustrated in Figure 5.3.

Here, the states represent the number of customers in SN_i. In Figure 5.3 the state dependent arrival rates $\lambda(n)$ (birth rates) are known, while we are interested in computing the state dependent service rates $\mu(n)$ (death rates). From the underlying CTMC of $\mathcal{AS}_i$, we can compute the conditional probabilities of having n customers in SN_i.

By formulating the local balance equations on the CTMC of the birth-death process in Figure 5.3 at w-th station (flow in = flow out):

$$\lambda(w-1)Pr(w-1) = \mu(w)Pr(w), \quad w = 1 \ldots K \tag{5.7}$$

From eq. (5.7) immediately follows

$$\mu(w) = \frac{Pr(w-1)}{Pr(w)}\lambda(w-1), \quad w = 1 \ldots K \tag{5.8}$$

which is just eq. (5.4) in slightly different notation. $\mu(w)$ is going to be the generator for the arrivals of $\mathcal{AS}_j$, i.e., it denotes the conditional throughput of SN_i.

As with many iterative methods, the uniqueness of the solution cannot be proven although numerical experience has shown that a unique fixed point does indeed exist, although convergence sometimes presents a problem [BD90].

For the case of $K = 1$ (independence of $M/G/1/1$ queue), equation (5.4) and (5.6) are equivalent to equation 4.3 for RTA, i.e., for $K = 1$ both methods are equivalent.

5.4 Delay Equivalence by Y. Li and M. Woodside

The work of Li and Woodside [LW91], presents an alternative approach to compute the approximate throughput for stochastic MGs. In their work, the original system is also split into subsystems and a delay equivalence (DE) criterion is used for throughput approximation. Their service rates for aggregated subsystems are marking dependent. In a very recent work [WL91], the authors present an application, while the computed service rates are made constant in order to get acceptable robustness.

Consider the two aggregated net systems in Figure 5.4 (called auxiliary systems in Woodside's paper). In $\mathcal{AS}_1$, a customer suffers two delays as he cycles through the system: a delay in SN_1 itself (D_1), and a delay through the aggregation of SN_2 (d_2 represented by q_{12} and τ_2 with firing rate μ_2). Similarly d_1 and D_2 represent the delays in $\mathcal{AS}_2$ (note that d represents the delay in the *aggregate* transitions τ, while D represents the delay in the unaggregated subsystems, i.e., SN).

Let K be the customer population (i.e., the number of tokens in the basic skeleton), $\mathrm{E}[q_{ij}]$ the expected number of tokens in place q_{ij}, $\mathcal{X}_i$ the throughput of $\mathcal{AS}_i$ and $\mu_i(l)$ the state dependent firing rate of τ_i when l customers are present in its input place q_{ji}. The delays can obtained by Little's law:

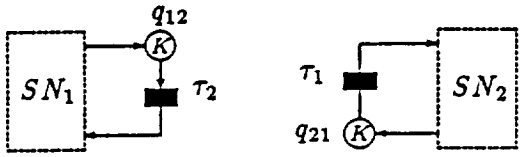


Figure 5.4: Aggregated (Auxiliary) Net Systems

$$D_1 = \frac{K - \mathrm{E}[q_{12}]}{\mathcal{X}_1} \tag{5.9}$$

$$d_2 = \frac{\mathrm{E}[q_{12}]}{\mathcal{X}_1} \tag{5.10}$$

$$D_2 = \frac{K - \mathrm{E}[q_{21}]}{\mathcal{X}_2} \tag{5.11}$$

$$d_1 = \frac{\mathrm{E}[q_{21}]}{\mathcal{X}_2} \tag{5.12}$$

The key to Woodside's method is, that in steady state the delay a token suffers in the subnet is the same as in the aggregate. For example, the delay of a customer in SN_1 of $\mathcal{AS}_1$ should be the same as in q_{21} and τ_1 in $\mathcal{AS}_2$. Therefore

$$D_1 = d_1 \tag{5.13}$$

$$D_2 = d_2 \tag{5.14}$$

5.4.1 State Dependent Firing Rates

Let us first consider the case when the aggregated transition has a state dependent firing rate [LW91]. The basic approach is to first compute the equivalent delay for $l = 1$ $(1/\mu(1))$ where l is the customer load. This information is used to compute $1/\mu(2)$ and successively up to $1/\mu(K)$.

(A) Case: $K = 1$

Here,

$$d_2 = \frac{\mathrm{E}[q_{12}]}{\mathcal{X}_1} = \frac{1}{\mu_2} \tag{5.15}$$

(from Little's law, $\mathcal{X}_1 = \mu_2 \, \mathrm{E}[q_{12}]$). Now with $d_2 = D_2$ and writing the equation in iterative form,

$$\mu_2^{(r+1)} = \frac{\mathcal{X}_2^{(r)}}{1 - \mathrm{E}[q_{21}]^{(r)}} = \frac{1}{D_2^{SN_2(r)}} \tag{5.16}$$

similarly for $\mu_1^{(r+1)}$,

$$\mu_1^{(r+1)} = \frac{\mathcal{X}_1^{(r)}}{1 - \mathrm{E}[q_{12}]^{(r)}} = \frac{1}{D_1^{SN_1(r)}} \tag{5.17}$$

Property 5.4.1 *For the particular case of a single customer, delay equivalence and response time approximation are equivalent.*

Proof: Let us consider $\mathcal{AS}_2$ at a given iteration: We know $\mathcal{X}_2$ and μ_1 and we have to determine μ_2. We can either solve equation (4.3) or simply write:

$$\frac{1}{\mathcal{X}_2} = \frac{1}{\mu_1} + \frac{1}{\mu_2} \tag{5.18}$$

rearranging terms yields

$$\mu_2 = \frac{\mathcal{X}_2 \, \mu_1}{\mu_1 - \mathcal{X}_2} \tag{5.19}$$

We have to show that equation (5.16) and (5.19) are the same, i.e.,

$$\frac{\mathcal{X}_2 \, \mu_1}{\mu_1 - \mathcal{X}_2} = \frac{\mathcal{X}_2}{1 - \mathrm{E}[q_{21}]} \tag{5.20}$$

From [GH85], we know that the steady state probability of an $M/G/c/c$ and an $M/M/c/c$ queue are the same if the means of the servers are the same. Therefore, we can write $\mathrm{E}[q_{12}]$ and $\mathcal{X}_2$ as if they were obtained by solving an $M/M/1/1$ queue.

$$\mathrm{E}[q_{12}] = \frac{\mu_2}{\mu_1 + \mu_2} \quad \text{and} \quad \mathcal{X}_2 = \frac{\mu_2 \mu_1}{\mu_1 + \mu_2} \tag{5.21}$$

By substituting eq. (5.21) into (5.20), the equality follows. ∎

Based on the previous results, the following property holds:

Property 5.4.2 *Because of the independence of the $M/1/G/1/1$ queue, for the case of a single customer ($K = 1$), RTA, Marie's method and delay equivalence yield the same results.*

(B) Case: $K > 1$

In the previous steps, the firing rates of τ for $n = 1 \ldots L-1$ customers have been determined. The present step is concerned with finding $\mu(L)$.

Substituting $\mathcal{X}_1 = \sum_{n=1}^{L} \mu_2(n) \cdot \Pr[m(q_{12}) = n]$ into equation (5.10)

$$d_2 = \frac{\mathrm{E}[q_{12}]}{\mu_2(L) \cdot \Pr[m(q_{12}) = L] + \sum_{n=1}^{L-1} \mu_2(n) \cdot \Pr[m(q_{12}) = n]} \tag{5.22}$$

As before, set $d_2 = D_2$ and by using eq. (5.11) the iterative equation follows:

$$\mu_2^{(r+1)}(L) = \frac{1}{\Pr^{(r)}[m(q_{12}) = L]} \left(\frac{\mathrm{E}^{(r)}[q_{12}]}{L - \mathrm{E}^{(r)}[q_{21}]} - \sum_{n=1}^{L-1} \mu_2(n) \cdot \Pr^{(r)}[m(q_{12}) = n] \right) \tag{5.23}$$

Similarly, $\mu_1^{(r+1)}(L)$ is given by;

$$\mu_1^{(r+1)}(L) = \frac{1}{\Pr^{(r)}[m(q_{21}) = L]} \left(\frac{\mathrm{E}^{(r)}[q_{21}]}{L - \mathrm{E}^{(r)}[q_{12}]} - \sum_{n=1}^{L-1} \mu_1(n) \cdot \Pr^{(r)}[m(q_{21}) = n] \right) \tag{5.24}$$

The iteration has to be performed for $L = 1 \ldots K$.

Note that for the computation of $\mu_2^{(r+1)}(N)$ the quantities $\Pr[m(q_{12}) = n]$, $n = 1 \ldots K$ and $\mathrm{E}[q_{12}]$ are computed with $\mathcal{AS}_1$ while $\mathrm{E}[q_{21}]$ is computed with $\mathcal{AS}_2$ (similarly for $\mu_1^{(r+1)}(N)$). At the current iteration, quantities from the analysis of *both* subnets are used.

5.4.2 State Independent Service Rate

We now examine DE for the case of state independent firing rates for the aggregated transitions [WL91].

Substituting $\mathcal{X}_1 = \mu_2 \cdot (1 - \Pr[m(q_{12}) = 0])$, into equation (5.10)

$$d_2 = \frac{\mathrm{E}[q_{12}]}{\mu_2 \cdot (1 - \Pr[m(q_{12}) = 0])} \tag{5.25}$$

As before, set $d_2 = D_2$ and by using eq. (5.11)

$$\frac{K - \mathrm{E}[q_{21}]}{\mathcal{X}_2} = \frac{\mathrm{E}[q_{12}]}{\mu_2 \cdot (1 - \Pr[m(q_{12}) = 0])} \tag{5.26}$$

Rewriting eq. (5.26) in iterative form yields

$$\mu_2^{r+1} = \frac{\mathrm{E}^r[q_{12}]}{K - \mathrm{E}^r[q_{21}]} \cdot \frac{\mathcal{X}_2^r}{(1 - \Pr^r[m(q_{12}) = 0])} \tag{5.27}$$

Similarly

$$\mu_1^{r+1} = \frac{\mathrm{E}^r[q_{21}]}{K - \mathrm{E}^r[q_{12}]} \cdot \frac{\mathcal{X}_1^r}{(1 - \Pr^r[m(q_{21}) = 0])} \tag{5.28}$$

As for the case of state dependent firing rates, quantities from the analysis of both subnets are used.

The iterative process is now straightforward. The structural decomposition is the same in all methods, only the computation of the new firing rate changes (step 5 in Algorithm 1: Quantitative Analysis of SISO cuts).

5.4.3 Comparison of Response Time Approximation and Delay Equivalence

In order to compare the two methods, it is useful to examine the solutions when the iteration has reached steady state, i.e., $\mathcal{X}_1 = \mathcal{X}_2$. From (5.10) and (5.11), we have

$$\mathrm{E}[q_{12}] = K - \mathrm{E}[q_{21}] \tag{5.29}$$

and the first term of eq. (5.27) disappears. In order to interpret the results, we will indicate with superscripts how the quantities are computed, e.g., $\mathcal{X}_1^{\mathcal{AS}_1}$ indicates that $\mathcal{X}_1$ is computed by solving the underlying CTMC of $\mathcal{AS}_1$. Using equation (5.29), (5.27) simplifies to:

$$\mu_2 = \frac{\mathcal{X}_2^{\mathcal{AS}_2}}{(1 - \Pr[m(q_{12}) = 0]^{\mathcal{AS}_1})} \tag{5.30}$$

In steady state, the above formula also holds for RTA, *except* that $\Pr[m(q_{12}) = 0]$ is computed based on the basic skeleton, while DE uses $\Pr[m(q_{12}) = 0]$ as computed with $\mathcal{AS}_1$. This derives from the fact that in RTA we reduce the subnets to the basic skeleton.

	use of $\mathcal{BS}$	use of $\mathcal{AS}_{1,2}$	Iterative	Information used
RTA	X	X	X	Throughput
Marie	-	X	X	Marking probabilities
Delay Equiv	-	X	X	Throughput and Marking probabilities
FEA	X	X*	-	* Throughput as a function of number of customers

Table 5.3: Comparison of the Stochastic Approximation Methods

5.5 Conclusions

FEA is computationally efficient, although its accuracy may be low due to the previously outlined problems. Its accuracy can be improved by aggregating only *one* subnet to a flow equivalent transition.

It has been shown that Marie's method can be used in the context of decomposition of MGs by a single cut because the basic skeleton is in product form.

DE works directly with the delay in the paths and does not reduce or make the abstraction to a basic skeleton. In this thesis we use a response time approximation based on the throughput of the aggregated net systems, i.e., we reduce our system to the basic skeleton (an $M/M/1/K$ queue). In steady state, the results using both methods are different.

For the case of a single customer, RTA, Woodside's method and Marie's method yield the same result due to the independence of the $M/G/1/K$ queue.

Table 5.3 summarizes the characteristics of the four presented methods.

CHAPTER 6

Examples for Marked Graph Systems

6.1 Introduction

In this chapter we give several numerical examples for the SISO and SIMO cuts introduced in the previous section. We compare the results obtained with single level and hierarchical Response Time Approximation (RTA), Flow Equivalent Aggregation (FEA), Marie's method and delay equivalence.

6.2 Fork/Join Queueing Network

The net system on top of the hierarchy in Figure 6.1 shows a Fork/Join QN with blocking. The exact value of its throughput is 0.22866, computed with the underlying CTMC which has 33480 states using the tool SPNP [DBCT85].

6.2.1 Single SISO Cut A

We define the SISO cut A through p_4 and p_5 (not shown in Figure6.1). We now follow the steps for the analysis outlined in Algorithm 1.

(1.a) The net is partitioned (Fig 6.1) by means of the canonical SISO cut ξ_A through p_4 and p_5. $\mathcal{AS}_1$ is defined by the transitions left of the cut, while $\mathcal{AS}_2$ is defined to the right of ξ_A.

(1.b) A solution for LPP3 (Section 4.3.2) is:
$M(p_1)=2, M(p_4)=3, M(p_6)=3, M(p_7)=2, M(p_8)=3, M(p_{12})=3, M(p_{13})=2.$
i.e., the customer load at cut A is $K=M(q_{21}=p_4)=3$.

(1.c) In order to show the rapid convergence of RTA (Section 4.3.3), we *arbitrarily* select the service rate of τ_2 as $\mu_1^0 = 20.0$.

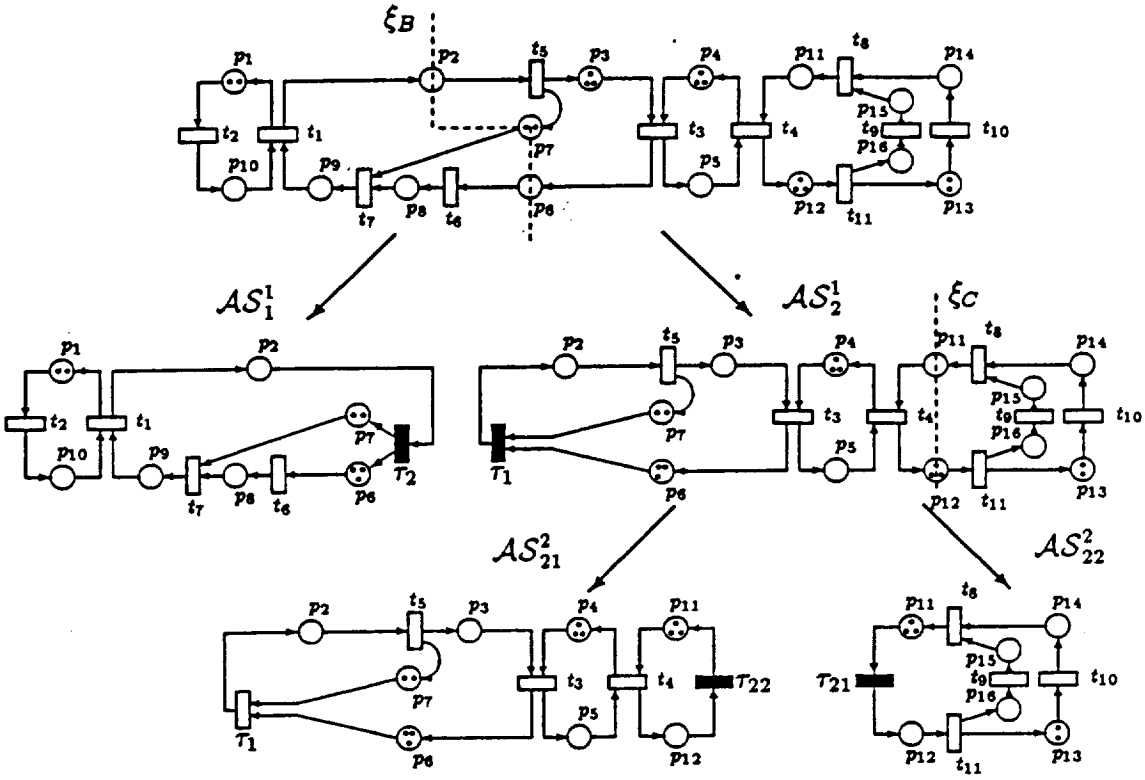


Figure 6.1: Fork/Join Queueing Network with Aggregated Net Systems Obtained from the Hierarchical Decomposition at Cut ξ_B and ξ_C (*Service Times*: $s_1 = 0.33$, $s_2 = 2$, $s_3 = 1$, $s_4 = 4$, $s_5 = 2$, $s_6 = 0.5$, $s_7 = 1$, $s_8 = 1$, $s_9 = 1$, $s_{10} = 1$,$s_{11} = 1$)

(2) The results of the iterations are shown in Table 6.1:

6.2.1.1 Response Time Approximation

In order to show the rapid convergence of the method, the iterative results are shown in Table 6.1. The initial value of $\mu_1^0 = 20.0$ is far off the final value of 0.59674.

The error of the throughput introduced by the aggregation is -0.14%, while $\mathcal{AS}_1$ has only 372 states and $\mathcal{AS}_2$ has 360 states. By finding proper initial values for the iteration (Algorithm 1, Step 3) the maximum cycle time in SN_1 is 1.278, therefore $\mu_1^0 = 0.7826$. This way, we reduce the number of CTMCs which have to be solved for the same precision by one.

$\mu_1^0 = 20.0;\ \mathcal{X}_{exact} = 0.22866$				
r	$\mathcal{X}_2^{r-1}$	μ_2^{r-1}	$\mathcal{X}_1^r$	μ_1^r
1	0.23281	0.23281	0.22444	0.60422
2	0.22852	0.23749	0.22824	0.59674
3	0.22833	0.23759	stop	

Table 6.1: Results Using RTA for SISO Cut A at Fork/Join QN

6.2.1.2 Flow Equivalent Aggregation

In this example the firing rate of $\mathcal{AS}_1$ is the same for all possible customer populations. Here, the merged interface places form a selfloop with τ_2, i.e., all tokens are trapped. Table 6.2 shows the conditional throughputs as a function of the number of tokens.

	$\mathcal{AS}_1$	$\mathcal{AS}_2$
conditional throughput	0.3430406	0.232809
tangible state space	93	90

Table 6.2: Results Using FEA for SISO Cut A Fork/Join QN

By substituting the values in Table 6.2 into the basic skeleton, the throughput is computed to be equal to 0.20313, i.e., the error introduced by the FEA is -11.2%.

6.2.1.3 Delay Equivalence

Table 6.3 shows the results using state independent delay equivalence aggregation (Section 5.3. The number of iterations required to reach convergence is higher than in RTA, while the accuracy is comparable (the introduced error is 0.18%).

6.2.2 SIMO/MISO Cut B

The resulting net systems are shown in Level 1 of the hierarchical decomposition presented in Figure 6.1 ($\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$). A solution for LPP5 (Rule 2 in Section

C-2

$\mu_1^0 = 1.0;\ \mu_2^0 = 1.0;\ \mathcal{X}_{exact} = 0.22866$				
r	$\mathcal{X}_1^r$	μ_1^r	$\mathcal{X}_2^r$	μ_2^r
1	0.342741	0.167135	0.146091	0.189616
2	0.186681	1.310073	0.232623	0.226793
3	0.219454	0.561616	0.227288	0.247681
4	0.236306	0.624189	0.228989	0.236558
5	0.227493	0.638746	0.229293	0.238451
6	0.229019	0.627703	0.229065	0.238784
7	0.229287	0.630124	0.229117	0.238509
8	0.229066	0.630256	stop	

Table 6.3: Results Using Delay Equivalence for SISO Cut A at Fork/Join QN

4.4) is:

$$M(p_1) = 2, M(p_5) = 3, M(p_7) = 2, M(p_6) = 3, M(p_{12}) = 3, M(p_{13}) = 2.$$

i.e., the customer load at the cut is: $K = 2$.

6.2.2.1 Response Time Approximation

For this cut, the error introduced in the throughput is 0.18%. $\mathcal{AS}_1^1$ has only 48 states, while $\mathcal{AS}_2^1$ has 3240 states. The large difference in the size of the respective state space is due to the fact that the cut is not balanced in the sense that it does not produce aggregated net systems with roughly the same state space (which is obviously bad from a computational point of view). Nevertheless, it has been considered for illustrative purposes. Table 6.4 shows the iteration results.

$\mu_1^0 = 20.0$				
r	$\mathcal{X}_2^{r-1}$	μ_2^{r-1}	$\mathcal{X}_1^r$	μ_1^r
1	0.23125	0.23122	0.68693	0.21317
2	0.25172	0.22919	0.67390	0.22851
3	0.25248	0.22909	0.67343	0.22906
4	0.25251	0.22908	stop	

Table 6.4: Results Using RTA for SIMO Cut B for Fork/Join QN

6.2.2.2 Flow Equivalent Aggregation

Similar to the previous example, the conditional throughput of $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$ (i.e., the firing rate for τ_1 and τ_2) is the same for all possible customer populations as shown in Table 6.5. The throughput obtained using FEA is equal to 0.19712, i.e., the error introduced by the aggregation is -13.79%.

		$\mathcal{AS}_1$	$\mathcal{AS}_2$
conditional throughput	$(n = 1)$	0.43619	0.22941
tangible state space	$(n = 1)$	15	1080
conditional throughput	$(n = 2)$	0.46801	0.23123
tangible state space	$(n = 2)$	27	1440

Table 6.5: Results Using FEA for SISO Cut B in Fork/Join QN

6.2.2.3 Delay Equivalence

Table 6.6 shows the results using state independent delay equivalence aggregation (Section 5.4) at cut B. The error introduced by the aggregation is 0.05%.

	$\mu_1(1) = 1.0$		$\mu_2(1) = 1.0$	
r	$\mathcal{X}_1^r$	μ_1^r	$\mathcal{X}_2^r$	μ_2^r
1	0.431732	0.347934	0.215488	0.254321
2	0.230400	0.740182	0.229558	0.241750
3	0.221138	0.632954	0.228680	0.254576
4	0.230585	0.639804	0.228756	0.251808
5	0.228572	0.641122	0.228770	0.252058
6	0.228755	0.640600	0.228765	0.252084

Table 6.6: Results Using Delay Equivalence for SISO Cut B at Fork/Join QN

6.2.2.4 Marie's Method

Table 6.7 shows the results obtained using Marie's method for both cuts. The error introduced by the aggregation is −0.09% for cut A and 0.23% for cut B.

All initial values 1.0		
r	$\mathcal{X}$(cut A)	$\mathcal{X}$(cut B)
1	0.193942	0.201954
2	0.236053	0.245348
3	0.227551	0.227600
4	0.228805	0.229971
5	0.228446	0.229053
6		0.229177

Table 6.7: Results Using Marie's Method for SISO Cut B at Fork/Join QN (throughput shown for the basic skeleton system)

6.2.3 Hierarchical Decomposition using RTA

Figure 6.1 also shows the hierarchical decomposition. The marking of the aggregated nets are obtained from Algorithm 2 (Section 4.6). Cut B partitions the original net system into the aggregated net systems $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$ at Level 1. On the next level, only $\mathcal{AS}_2^1$ is partitioned by C to form $\mathcal{AS}_{21}^2$ and $\mathcal{AS}_{22}^2$. $\mathcal{X}_2$ is computed at Level 2 using $\mathcal{AS}_{21}^2$ and $\mathcal{AS}_{22}^2$. The initial rate for μ_2^0 was chosen as 20.0. The iteration results are shown in Table 6.8.

$\mu_1^0 = 20.0$				
r	$\mathcal{X}_2^{r-1}$	μ_2^{r-1}	$\mathcal{X}_1^r$	μ_1^r
1	0.46735	0.46760	0.22376	0.27148
2	0.24255	0.66209	0.23042	0.25512
3	0.23097	0.67181	0.23058	0.25459
4	0.23059	0.67199	stop	

Table 6.8: Results Using Hierarchical Decomposition at Fork/Join QN

The original net system has 33480 states, while $\mathcal{AS}_1^1$ has 48 states, $\mathcal{AS}_{21}^2$ has 144 states, and $\mathcal{AS}_{22}^2$ has 90 states. The error introduced during the aggregation is 0.84%. During the iterative process, $\mathcal{AS}_1^1$ was analyzed a total of 4 times while $\mathcal{AS}_{21}^2$ and $\mathcal{AS}_{22}^2$ were analyzed 7 times.

6.2.4 Comparison

Table 6.9 shows a summary of all results for the throughput of the Fork/Join QN, as well as the largest state space of an aggregated subnet encountered during the approximation.

original net: $\mathcal{X} = 0.22866$, State Space = 33480				
Cut A				
Method	Throughput	Error	State Space	Iterations
RTA	0.22833	−0.14%	372	3
FEA	0.20313	−11.16%	93	—
Marie's Method	0.22845	−0.09%	372	5
Delay Equivalence	0.22907	0.18%	372	8
Cut B				
RTA	0.22908	0.18%	3240	4
FEA	0.20313	−13.79%	1440	—
Marie's Method	0.22918	−0.23%	3240	6
Delay Equivalence	0.22907	0.05%	3240	6
Hierarchical RTA	0.23059	0.84%	144	4

Table 6.9: Comparison of the Approximation Results for Fork/Join QN

With the exception of FEA, all methods show very good accuracy at a significant computational saving. RTA shows the fastest convergence.

6.3 Four Task Pipeline

We now analyze the MG in Figure 4.7 whose *structure* (but not the marking) represents a four task pipeline found in concurrent software environments. The example is inspired by the example in Y. Li's paper [LW91]. It is analyzed with two different initial markings:

$M_0^{(1)}$: This is the marking shown in Figure 4.7. The exact value of the throughput for $M_0^{(1)}$ is equal to 0.513195, computed with the underlying CTMC which has 4500 states (using the tool SPNP [DBCT85]). Note that p_{20}, p_{21} and p_{22} are implicit places for the initial marking shown.

$M_0^{(2)}$: Now $M(p_{21})=1$, $M(p_4)=1$ and $M(p_{19})=2$, while all other markings remain the same. None of the places in the net is now implicit. The exact throughput of this MG system is 0.490362, while the underlying CTMC has 2916 states.

The service times are as follows:
$s_1 = 0.20$, $s_2 = 0.20$, $s_3 = 0.10$, $s_4 = 0.10$, $s_5 = 1.00$, $s_6 = 0.10$, $s_7 = 0.20$, $s_8 = 1.20$, $s_9 = 0.10$, $s_{10} = 0.09$, $s_{11} = 0.90$, $s_{12} = 0.08$, $s_{13} = 0.07$, $s_{14} = 0.15$, $s_{15} = 0.10$, $s_{16} = 1.00$.

6.3.1 Single SISO Cut B

We define the SISO cut B through p_7 and p_{17} as shown on top of the hierarchy in Figure 4.7. The aggregated net systems are shown in the first level of the decomposition ($\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$). The net in Figure 4.7 is already shown with a solution for LPP3, i.e., the number of tokens in q_{21} ($q_{21} = p_{17}$, Figure 4.7) has already been maximized. The customer load at cut B is two.

6.3.2 SISO Cut B, Response Time Approximation

In order to examine the speed of convergence of RTA as a function of the initial values of the iteration, we study the system for different values μ_2^0 of τ_2 in

	Initial Marking $M_0^{(1)}$				Initial Marking $M_0^{(2)}$			
	Iteration Results for $\mu_2^0=0.1$							
r	$\mathcal{X}_1^{r-1}$	μ_1^{r-1}	$\mathcal{X}_2^r$	μ_2^r	$\mathcal{X}_1^{r-1}$	μ_1^{r-1}	$\mathcal{X}_2^r$	μ_2^r
1	.098578	.784181	.557147	0.89741	.098764	.845246	.512319	.707871
2	.481131	.615319	.503148	1.03136	.471021	.705199	.485372	.753004
3	.498699	.607624	.499849	1.03956	.482587	.697578	.483479	.755988
4	.499609	.607214	stop		.483300	.697094	.483357	.756180
	Iteration Results for $\mu_2^0=1.0$							
1	.495060	.609250	.500553	1.03780	.525254	.664642	.474687	.769652
2	.499416	.607301	.499708	1.03990	.486492	.694906	.482804	.757049
3	.499647	.607197	stop		.483553	.696922	.483314	.756248
4					.483362	.697052	stop	
	Iteration Results for $\mu_2^0=10.0$							
1	.570988	.572760	.483818	1.07983	.577380	.579211	.446575	.811949
2	.503844	.605292	.498832	1.04209	.495621	.688439	.481142	.759651
3	.499886	.607089	.499616	1.04013	.484168	.696502	.483208	.756414
4	.499673	.607186	stop		.483402	.697025	.483340	.756207

Table 6.10: Results Using RTA for SISO cut B with Different Initial Values of τ_2 at Pipeline System

$\mathcal{AS}_1^1$. Table 6.10 shows the iterative results for the two initial markings. The error of the throughput introduced by the aggregation for $M_0^{(1)}$ is -2.71%, while $\mathcal{AS}_1^1$ has 75 states and $\mathcal{AS}_2^1$ has 180 states. For $M_0^{(2)}$ the error of the throughput introduced by the aggregation is -1.45%, while $\mathcal{AS}_1^1$ has 81 states and $\mathcal{AS}_2^1$ has 108 states.

From the results of the different initial values in Table 6.10 we see that initial values "close" to the final ones can reduce the number of iterations by one.

6.3.3 SISO Cut B, Flow Equivalent Aggregation

Table 6.11 shows the conditional throughputs obtained with all possible customer populations.For $M_0^{(1)}$, the throughput obtained using FEA is equal to 0.38897, therefore the error introduced by the aggregation is -24.2%. For $M_0^{(2)}$, the approximated throughput is 0.37797 with an error of -22.91%.

Initial Marking $M_0^{(1)}$		$\mathcal{AS}_1^1$	$\mathcal{AS}_2^1$
conditional throughput	$(n=1)$	0.50267	0.62190
tangible state space	$(n=1)$	20	60
conditional throughput	$(n=2)$	0.57433	0.62190
tangible state space	$(n=2)$	50	60
Initial Marking $M_0^{(2)}$		$\mathcal{AS}_1$	$\mathcal{AS}_2$
conditional throughput	$(n=1)$	0.57742	0.55699
tangible state space	$(n=1)$	36	36
conditional throughput	$(n=2)$	0.57742	0.55699
tangible state space	$(n=2)$	36	36

Table 6.11: Results Using Flow Equivalent Aggregation for SISO Cut B at Pipeline System

6.3.4 SISO Cut B, Marie's Method

Table 6.12 shows the results obtained by using Marie's method for both initial markings. All initial values were taken to be one.

All initial values 1.0		
r	$\mathcal{X}(M_0^{(1)})$	$\mathcal{X}(M_0^{(2)})$
1	0.455636	0.428801
2	0.519786	0.508209
3	0.499322	0.479429
4	0.505794	0.490432
5	0.503682	0.486397
6	0.504346	0.487948
7	0.504129	0.487379

Table 6.12: Results Using Marie's Method for SISO Cut B at Pipeline System (Throughputs shown for the Basic Skeleton System)

The error introduced by the aggregation is -1.77% for $M_0^{(1)}$ and -0.61% for $M_0^{(2)}$. Convergence did not present a problem.

6.3.5 SISO Cut B, Delay Equivalence

In Table 6.13 the iteration Results Using *state dependent* delay equivalence with the initial marking $M_0^{(1)}$ are presented. The values of μ_i are now a function of the number of customers n in its input place. In a previous step, the firing rates with a costumer load equal to one were determined to be: $\mu_1(1) = 1.00010$ and $\mu_2(1) = 0.60499$. These values were also used as the initial values for $\mu_1^0(2)$ and $\mu_2^0(2)$ respectively. The second (fourth) column shows the throughput obtained with $\mathcal{AS}_1$ ($\mathcal{AS}_2$) at the current iteration. In the third (fifth) is the new firing rate of $\mu_2(2)$ ($\mu_1(2)$) which is going to be used in the next iteration in $\mathcal{AS}_1$ ($\mathcal{AS}_2$).

	$\mu_1(1) = 1.00010$		$\mu_2(1) = 0.60499$	
r	$\mathcal{X}_1^r$	$\mu_2^r(2)$	$\mathcal{X}_2^r$	$\mu_1^r(2)$
1	0.495072	1.214634	0.498700	0.623275
2	0.506690	1.122975	0.505133	0.617402
3	0.502279	1.100289	0.503109	0.611861
4	0.501071	1.129935	0.501163	0.616615
5	0.502640	1.135421	0.502834	0.615856
6	0.502921	1.116811	0.502569	0.615088
7	0.501956	1.127689	0.502300	0.615153
8	0.502524	1.125427	0.502323	0.615761
9	0.502407	1.126214	0.502536	0.615194

Table 6.13: Results Using State Dependent Delay Equivalence for SISO Cut B and $M_0^{(1)}$ at Pipeline System

The error introduced by the approximation is -2.1%. Note that the initial values of $\mu_1(2)$ and $\mu_2(2)$ are already very close to their final values, still it takes 9 steps to achieve convergence. If initial values for $\mu_2^0(2) = \mu_1^0(2) = 1.0$ are used, the method takes 29 steps to converge. For the same net structure, but for different initial markings, we were not able to find initial values for which state dependent delay equivalence converges. This method is not robust, i.e., convergence is depends very much on the initial marking and is often difficult to achieve.

Table 6.14 shows the results for state independent delay equivalence. For $M_0^{(1)}$, the introduced error is -2.44% (-2.49% for $M_0^{(2)}$).

$\mu_1^0 = 1.0$		$\mu_2^0 = 1.0$		
Initial Marking $M_0^{(1)}$				
r	$\mathcal{X}_1^r$	$\mu_2^r(2)$	$\mathcal{X}_2^r$	$\mu_1^r(2)$
1	0.495060	0.530136	0.461693	1.243986
2	0.517779	0.609419	0.500626	1.016688
3	0.497029	0.613985	0.502582	1.048344
4	0.500566	0.608701	0.500315	1.051924
5	0.500950	0.609637	0.500720	1.049439
6	0.500684	0.609703	stop	
Initial Marking $M_0^{(2)}$				
r	$\mathcal{X}_1^r$	$\mu_2^r(2)$	$\mathcal{X}_2^r$	$\mu_1^r(2)$
1	0.525254	0.543843	0.432350	0.801388
2	0.493443	0.699546	0.483973	0.705709
3	0.470429	0.680914	0.479159	0.740314
4	0.479483	0.675243	0.477630	0.736225
5	0.478459	0.678178	0.478425	0.735028
6	0.478157	0.677693	0.478294	0.735662

Table 6.14: Results Using State Independent Delay Equivalence for SISO Cut B and $M_0^{(1)}$ at Pipeline System

6.3.6 Hierarchical Decomposition

Figure 4.7 shows a hierarchical decomposition by the cuts A, B and C. The marking of the aggregated nets are obtained from the procedure outlined in Algorithm 2 in Section 4.6 (with the shown marking, the number of tokens in all interface places is already maximized). Cut B partitions the original net system into the aggregated net systems $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$ at Level 1. On the next level, $\mathcal{AS}_1^1$ is partitioned by A to form $\mathcal{AS}_{11}^2$ and $\mathcal{AS}_{12}^2$, while $\mathcal{AS}_2^1$ is partitioned by C to form $\mathcal{AS}_{21}^2$ and $\mathcal{AS}_{22}^2$. $\mathcal{X}_1$ ($\mathcal{X}_2$) is computed at Level 2 using $\mathcal{AS}_{11}^2$ and $\mathcal{AS}_{12}^2$ ($\mathcal{AS}_{21}^2$ and $\mathcal{AS}_{22}^2$). The iteration results are shown in Table 6.15, while a comparison of the state space and

the number of iterations is shown in Table 6.16. All initial rates were chosen to be 10.0.

Initial Marking $M_0^{(1)}$				
r	$\mathcal{X}_1^{r-1}$	μ_1^{r-1}	$\mathcal{X}_2^r$	μ_2^r
1	0.548352	0.5499239	0.466712	1.056116
2	0.477591	0.5672174	0.474982	1.034340
3	0.475409	0.5679352	0.475307	1.033512
Initial Marking $M_0^{(2)}$				
r	$\mathcal{X}_1^{r-1}$	μ_1^{r-1}	$\mathcal{X}_2^r$	μ_2^r
1	0.568095	0.5698403	0.441979	0.8121824
2	0.486450	0.666298	0.473333	0.7622874
3	0.476178	0.6735495	0.475281	0.7590964
4	0.475474	0.6740473	0.475411	0.7588733

Table 6.15: Results Using Hierarchical Decomposition for Pipeline System

Tangible State Space				
marking	$\mathcal{AS}_{11}^1$	$\mathcal{AS}_{12}^2$	$\mathcal{AS}_{21}^2$	$\mathcal{AS}_{22}^2$
$M_0^{(1)}$	5	30	45	12
$M_0^{(2)}$	9	27	27	12
Number of Evaluations				
marking	$\mathcal{AS}_{11}^1$	$\mathcal{AS}_{12}^2$	$\mathcal{AS}_{21}^2$	$\mathcal{AS}_{22}^2$
$M_0^{(1)}$	6	4	7	4
$M_0^{(2)}$	10	10	10	7

Table 6.16: Tangible State Space for Hierarchical Decomposition and Number of Evaluations

From Table 6.16 we see that the state space is reduced by more than two orders of magnitude: For the original marking, the error introduced during the aggregation is -7.3% (-3.13% for the modified initial marking).

6.3.7 Comparison

Table 6.17 shows a summary of the results. The iterative aggregation methods introduce a very small error into the final result, and at very reasonable computational efforts. The only method that poses major convergence problems is state dependent delay equivalence, which often fails to converge. RTA converges particularly fast, even when using "bad" initial values for the iteration. FEA introduces an unacceptably large error in the aggregation. This is due to the strong dependence of the mean completion time on the interarrival process.

Initial Marking $M_0^{(1)}$				
Method	Throughput	Error	State Space	Iterations
original net	0.51319	—	4500	—
RTA	0.49967	−2.71%	180	3
FEA	0.38897	−22.91%	60	—
Marie's Method	0.50413	−1.77%	180	7
Delay Equivalence (state dependent)	0.50253	−2.1%	180	9
Delay Equivalence (state independent)	0.50068	−2.44%	180	6
Initial Marking $M_0^{(2)}$				
original net	0.49036	—	2916	—
RTA	0.48334	−1.45%	108	4
FEA	0.37797	−24.20%	36	—
Marie's Method	0.48738	−0.61%	108	7
Delay Equivalence (state independent)	0.478294	−2.49%	108	6

Table 6.17: Comparison of the Approximation Results for Pipeline System

6.4 Kanban System

The following is an application taken from the field of manufacturing systems. Figure 6.2 shows a 6 stage *Kanban system.*

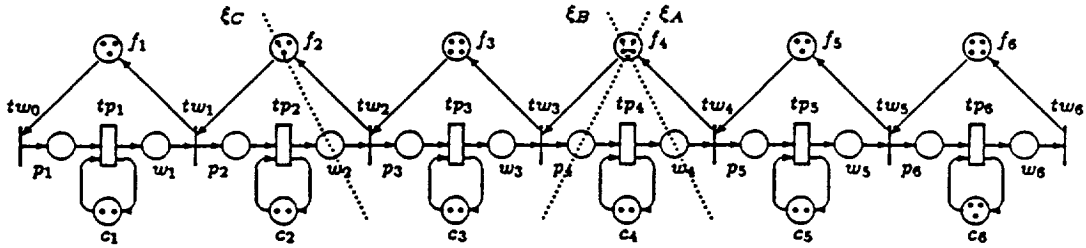


Figure 6.2: Six Stage Kanban System

The *just-in-time* philosophy for the control of manufacturing systems consists of producing just the needed parts at each production stage in just the right time. *Kanban* control is a way to implement a just-in-time manufacturing system. A Kanban is a ticket that accompanies a part through several stages of the production system. When a part of a given stage is consumed by the succeeding stage, the ticket is sent back to trigger the production of a new part. Petri nets have been shown to be well adapted to provide a unified modeling of decision-free Kanban systems [MFDD89] and most models can easily be represented by MGs.

In the MG system of Figure 6.2, places f_i ($i = 1 \ldots 6$) represent the available space in the work area (number of Kanban tickets), p_i the parts awaiting service, w_i the parts which have finished processing at transitions tp_i and c_i the available number of machines at each stage. In this MG system, we assume that a transition t enabled K times in a marking M (i.e., $K = \max\{k \mid M \geq k\ Pre[t]\}$) works at conditional speed K times it would work in the case it was enabled only once (*infinite server semantics*) [RP84, Zub85, HV85].

Here c_i limits the enabling degree of tw_i, i.e., tw_i is enabled $\min\{M(p_i), M(c_i)\}$ times. For the sake of simplicity we assume that all machines have exponentially distributed processing times with unity mean. The exact throughput is 1.496715, computed with the underlying CTMC which has a total of 93843 states (18040 vanishing and 75803 tangible markings using the tool GreatSPN [Chi85]).

For single level decomposition, the net is split into two subnets by a single SISO cut ξ_A defined through the *interface places* p_4 and f_4. As before, $\mathcal{AS}_1$ is defined to the left of the cut while $\mathcal{AS}_2$ is defined to the right of ξ_A. By solving LPP3 (Section 4.3.2, we find that a possible solution for the marking of the interface places is: $M(f_4) = 4$; $M(p_4) = 0$, i.e., the customer population is $K = M(q_{21} = f_4) = 4$.

6.4.1 Response Time Approximation

Table 6.18 shows the results using RTA. $\mathcal{AS}_1$ has a total of 1285 states (380 vanishing and 905 tangible markings) while $\mathcal{AS}_2$ has a total of 335 states (210 vanishing and 125 tangible markings). The introduced error is -1.47%.

The reduction is by a factor of 73 (=93843/1285) when the total state space is considered and by a factor of 47 (=18040/380) if only the tangible state space is considered.

$\mu_1^0 = 10.0$				
r	$\mathcal{X}_2^{r-1}$	μ_2^{r-1}	$\mathcal{X}_1^r$	μ_1^r
1	1.573246	1.574060	1.308050	1.706777
2	1.480033	2.058081	1.471149	1.690805
3	1.474889	2.072899	1.474685	1.690446

Table 6.18: Results Using RTA for Single Cut at Kanban System

6.4.2 Flow Equivalent Aggregation

Similar to the previous example, the throughput of $\mathcal{AS}_1$ is the same for all possible customer populations. Here, the merged interface places form a selfloop with τ_2, i.e., all tokens are trapped. Table 6.19 shows the conditional throughputs as a function of the number of tokens.

The throughput obtained using FEA is equal to 1.26180, i.e., the error introduced by the aggregation is -15.70%.

Conditional throughput $\mathcal{AS}_1 = 1.573250$ Number of tangible States = 50, $n=1,2,3,4$		
n	conditional throughput $\mathcal{AS}_2$	State Space $\mathcal{AS}_2$
1	0.956173	34
2	1.587574	42
3	1.652232	50
4	1.697872	58

Table 6.19: Results Using Flow Equivalent Aggregation at Kanban System

6.4.3 Marie's Method

Table 6.20 shows the results obtained by using Marie's method (Section 5.3). All initial values were taken to be one.

All initial values 1.0	
r	$\mathcal{X}$
1	1.839026
2	1.408609
3	1.523192
4	1.484177
5	1.495074
6	1.491380
7	1.492412
8	1.492063

Table 6.20: Results Using Marie's Method at Kanban System (Throughput Shown for the Basic Skeleton System)

The error introduced by the aggregation is -0.31%. Convergence did not present a problem.

6.4.4 Delay Equivalence

Table 6.21 show the results using state independent delay equivalence aggregation (Section 5.4). For this example, convergence depends on the initial values of

the iteration, e.g., using $\mu_1^0 = \mu_2^0 = 1.0$ delay equivalence does not converge. The number of iterations required is significantly higher than in RTA, while the accuracy is comparable (the introduced error is -0.34%).

In RTA and delay equivalence, the structure and marking of the aggregated net systems are the same, therefore the state space is also identical. Delay equivalence uses the conditional probabilities of customers in *both* aggregated net systems at the current iteration. We believe that this makes delay equivalence less robust, as an unfortunate combination of initial values can cause unstable oscillations.

	$\mu_1^0 = 1.0$		$\mu_2^0 = 4.0$	
r	$\mathcal{X}_1^r$	μ_1^r	$\mathcal{X}_2^r$	μ_2^r
1	0.995363	28.328484	1.694930	1.685173
2	1.473015	1.223924	1.113728	2.283125
3	1.560674	1.776035	1.388686	1.461586
4	1.370634	4.147622	1.650823	1.673865
5	1.469159	1.718533	1.367792	1.979070
6	1.537082	1.854180	1.414690	1.646106
7	1.459144	2.740666	1.579719	1.689514
8	1.474462	2.038670	1.466410	1.830199
9	1.512249	1.973474	1.449564	1.727731
10	1.486418	2.313612	1.523348	1.713405
11	1.482097	2.157780	1.493647	1.769801
12	1.498054	2.067290	1.473355	1.748789
13	1.492434	2.183312	1.498943	1.731198
14	1.487435	2.171140	1.496441	1.750136
15	1.492805	2.118218	1.485076	1.749755
16	1.492700	2.149813	1.491957	1.740347
17	1.490069	2.160998	1.494324	1.745181
18	1.491431	2.139305	1.489700	1.747555
19	1.492092	2.144376	1.490793	1.743924
20	1.491079	2.152818	1.492596	1.744530
21	1.491248	2.146035	1.491149	1.746062
22	1.491677	2.145147	stop	

Table 6.21: Results Using Delay Equivalence at Kanban System

6.4.5 Hierarchical Decomposition using RTA

Figure 6.2 also shows the cuts leading to a hierarchical decomposition. The markings of the aggregated nets are obtained from Algorithm 2 (Section 4.6). Cut B partitions the original net system into the aggregated net systems $\mathcal{AS}_1^1$ and $\mathcal{AS}_2^1$ at Level 1. On the next level, only $\mathcal{AS}_1^1$ is partitioned by C to form $\mathcal{AS}_{11}^2$ and $\mathcal{AS}_{12}^2$. $\mathcal{X}_1$ is computed at Level 2 using $\mathcal{AS}_{11}^2$ and $\mathcal{AS}_{12}^2$. The initial rate for μ_2^0 was chosen as 20.0. The iteration results are shown in Table 6.22.

$\mu_2^0 = 20.0$				
r	$\mathcal{X}_1^{r-1}$	μ_1^{r-1}	$\mathcal{X}_2^r$	μ_2^r
1	1.49408	1.49467	1.42838	2.69747
2	1.45576	1.52824	1.45317	2.66603
3	1.45433	1.52968	1.45422	2.66471

Table 6.22: Results Using Hierarchical Decomposition at Kanban System

Here, the state space is reduced by more than two orders of magnitude: $\mathcal{AS}_{11}$ has a 22 states, $\mathcal{AS}_{12}$ has 230 states while $\mathcal{AS}_2$ has 58 tangible states. The error introduced during the aggregation is -2.84%. The aggregated net systems have to be analyzed repetitively, e.g., during the iterative process, $\mathcal{AS}_2^1$ was analyzed a total of 3 times while $\mathcal{AS}_{11}^2$ and $\mathcal{AS}_{12}^2$ were analyzed 6 times.

6.4.6 Comparison

Table 6.23 shows a summary of the results.

With the exception of FEA, all methods show good accuracy. RTA provides a significant computational saving by showing the fastest convergence. In this example, the initial values for delay equivalence have to be chosen carefully in order to achieve convergence. The number of required iterations for this method is high when compared to RTA.

original net: $\mathcal{X} = 1.496715$, Tang. State Space = 75803				
Method	Throughput	Error	State Space	Iterations
RTA	1.47469	−1.47%	380	3
FEA	1.26180	−15.70%	58	—
Marie's Method	1.49206	−0.31%	380	7
Delay Equivalence	1.491677	−0.34%	380	22
Hierarchical RTA	1.45422	−2.84%	230	3

Table 6.23: Comparison of the Approximation Results for Kanban System

6.5 Conclusions

In this chapter we have presented numerical results for the approximation methods which were introduced in the previous chapter. For the particular examples, we can draw the following conclusions:

- FEA has the lowest state space of the CTMC and is therefore computationally very efficient. The drawback is obvious: in the examples, there exists a strong dependence of the service time on the arrival process, therefore as outlined in Section 5.4.3. the accuracy can suffer.

- Delay equivalence shows good accuracy, provided it converges. The choice of proper initial values is critical for convergence. This is particularly true for *state dependent delay equivalence*, which failed to converge in almost all cases. This method uses the token distribution of both aggregated net systems at the current iteration. In the initial step, these distributions have to be determined with the initial values of the iteration. A poor choice will lead to instabilities (and eventually non-convergence) in the iteration.

- RTA shows similar accuracy as delay equivalence, but at a greatly reduced computational cost. RTA is insensitive with respect to the initial values and is the one which requires the least amount of iterations. RTA does not make use

of the token distribution in the subnets, and only uses the throughput. We believe, that this accounts for the good robustness of the method. Hierarchical decomposition can possibly lead to a larger error.

- Marie's method shows the best accuracy for all SISO cuts, with a reasonable computational cost (about twice as many iterations as RTA). Note that Marie's method uses all information about the token distribution in the aggregated net systems, i.e., it uses more information than RTA. On the other hand, convergence is slower.

The savings in the state space immediately leads to a great saving in the time required to obtain the solution. For the Kanban example, the time needed on a SUN 4/330 SPARC with 16 Mb RAM to compute the exact solution of the original system was around 2 h. 30 m while the approximate solution computed with 3 iterations was obtained in less than 1 minute using SPNP [DBCT85].

At this point it is important to state that a Kanban system with 8 stages generates such a large state space that the computation of the exact throughput is impossible because of the lack of memory. Its decomposition into two aggregated nets of 4 stages plus the environment transitions (τ_1 and τ_2) leads to a computation in less than two minutes!

CHAPTER 7
Macroplace/Macrotransition Nets

7.1 Introduction

In the previous chapters we have introduced approximation methods for stochastic MGs. The modeling power of this net subclass is restricted, because decisions are forbidden. In this chapter, we introduce a new class of Petri nets, Macroplace-Macrotransition-nets (MPMT-nets), a subclass which allows limited choice, concurrency and sharing of resources [DJS92]. MPMT-nets have a greatly increased modeling power over SMs and MGs by iteratively combining Macrotransitions (where a single transition has been expanded to a MG) and Macroplaces (where a place has been refined to a SMs). By exploring the *qualitative properties* of MPMT-nets, we show that the previously introduced approximation technique for SISO cuts are applicable to this subclass of PNs.

7.2 MPMT-nets: Motivation and Definition

Even if some practical systems can be modeled by Fork/Join queueing networks with blocking or closed queueing networks (thus essentially strongly connected marked graphs state machines, respectively), the interleaving of choices and synchronization is fundamental in many cases. For example, the structure of flow line models with unreliable machines are neither MGs, nor SMs (e.g., see Figure 8.2).

A traditional way of interleaving choices and synchronization in a controlled way is given by *Free Choice nets*. The behavioral and structural analysis of FC nets is particularly elegant and well understood (see [ES90a] for a survey). Nevertheless, classical schemes in manufacturing in which two or more processes share a common

resource (e.g., a robot) cannot be modeled with FC net systems.

Here we introduce a different combination of the basic structures of SMs and MGs. The practical modeling with this new subclass of net systems has some advantages in the field of manufacturing with respect to the use of the well known subclass of live and bounded free choice net systems allowing e.g., the limited sharing of resources and some synchronization at choice places.

The new net subclass is called *MPMT-nets*, because they can be obtained by using the *macroplace* (MP) and *macrotransition* (MT) reduction/refinement rules. The MP-rule is strongly related to the concept of SMs [Sil81], while the MT-rule is its reverse-dual and therefore, strongly related to MGs (see [ES90a] for an integrated consideration of both rules).

The MP (MT) rule consists of the substitution of a given SM (MG) subnet by a single place (transition). Only a particular case of MP (MT) reduction rule is considered here: a unique way-in place (transition) and a unique way-out place (transition) exists.

Way-in places are those that can be used to "enter" into the subnet, and way-out places are those through which we can "leave" it. More formally:

Let $\mathcal{N}' = (P', T', F')$ be a subnet of $\mathcal{N} = (P, T, F)$
[i.e., $F' = F \cap ((P' \times T') \cup (T' \times P'))$]:

(a) $p' \in P'$ is a way-in place of $\mathcal{N}'$ iff ${}^\bullet p' \cap (T \setminus T') \neq \emptyset$

(b) $p' \in P'$ is a way-out place of $\mathcal{N}'$ iff $p'^\bullet \cap (T \setminus T') \neq \emptyset$

where $(T \setminus T')$ means the set T minus the set T'.

Way-in and way-out transitions are defined analogously.

Definition 7.2.1 *Let $\mathcal{N}'$ be a subnet of $\mathcal{N}$. $\mathcal{N}' = (P', T', F')$ is reducible to a place if:*

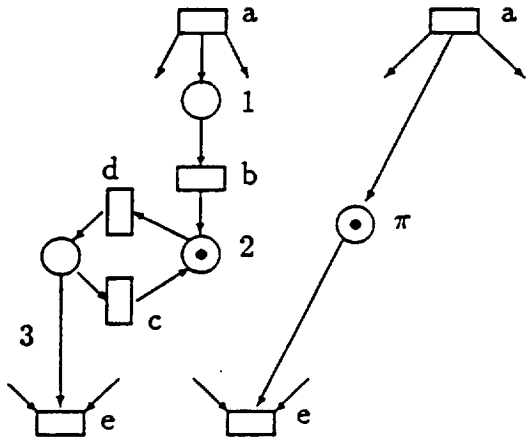


Figure 7.1: Macroplace Reduction Rule

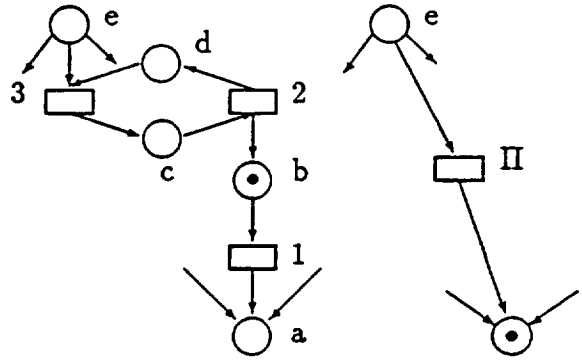


Figure 7.2: Macrotransition Reduction Rule: Disregarding the Marking, the Subnet is the Reverse-dual of that in Figure 7.1

(a) $\mathcal{N}'$ *is a state machine containing one way-in place,* p'_i*, and one way-out place,* p'_0 *(*$p'_i = p'_0$ *is possible).*

(b) For every $p' \in P'$*, there exists at least an* F'*-path from:*

(b.1) the way-in place p'_i *to* p'*.*

(b.2) p' *to the way-out place,* p'_0*.*

The next definition expresses the standard notion of substitution of a net by a place, π. See Figure 7.1 for an example.

Definition 7.2.2 *Let $\langle \mathcal{N}, M_0 \rangle$ be a system and $\mathcal{N}' = (P', T', F,)$ a subnet of $\mathcal{N} = (P, T; F)$ reducible to a place. The net $\mathcal{N}_r = (P_r, T_r; F_r)$, with:*

- $P_r = (P \setminus P') \cup \{\pi\}$
- $T_r = (T \setminus T')$
- $F_r = (F \cap ((P_r \times T_r) \cup (T_r \times P_r))) \cup F_\pi$, *where:*
  - $(t, \pi) \in F_\pi$ *iff* $(t, p_i') \in F$
  - $(\pi, t) \in F_\pi$ *iff* $(p_0', t) \in F$

is a macroplace reduction of $\mathcal{N}$, and π is the macroplace that replaces $\mathcal{N}'$. The system $\langle \mathcal{N}_r, M_r \rangle$ where $\mathcal{N}_r$ is the reduction of N and M_r is given by:

- $M_r(p) = M_0(p)$ *if* $p \neq \pi$
- $M_r(\pi) = \sum_{p \in P'} M_0(p)$

is called a macroplace reduction of $\langle \mathcal{N}, M_0 \rangle$.

The value of the macroplace concept lies in the following result [Sil81]: *The reduction of a macroplace preserves liveness and the bound of the places of the system.*

Moreover, in our particular case (single way-in/single way-out place), reversibility is also preserved. Therefore:

Property 7.2.1 *The reduction of a macroplace preserves liveness, the bound of places (thus boundedness) and reversibility.*

At the structural level, the reverse-dual (i.e., reversing the arcs, changing places to transitions and vice versa) of the MP-rule is the MT-rule.

Definition 7.2.3 *Let $\mathcal{N}'$ be a subnet of $\mathcal{N}$. $\mathcal{N}' = (P', T'; F')$ is reducible to a transition if:*

(a) $\mathcal{N}'$ is a marked graph containing one way-in transition, t'_i and one way-out transition, t'_0 ($t'_i = t'_0$ is possible).

(b) For every $t' \in T'$ there exists at least an F'-path from:

(b.1) the way-in transition, t'_i, to t'.

(b.2) t' to the way-out transition, t'_0.

Definition 7.2.4 *Let $\langle N, M_0 \rangle$ be a system and $\mathcal{N}' = (P', T'; F')$ a subnet of $\mathcal{N} = (P, T; F)$ reducible to a transition. The net $\mathcal{N}_r = (P_r, T_r; F_r)$ with:*

- $P_r = (P \setminus P')$
- $T_r = (T \setminus T') \cup \{\theta\}$
- $F_r = (F \cap ((P_r \times T_r) \cup (T_r \times P_r))) \cup F_\theta$, *where:*
  - $(p, \theta) \in F_\theta$ *iff* $(p, t'_i) \in F$
  - $(\theta, p) \in F_\theta$ *iff* $(t'_0, p) \in F$

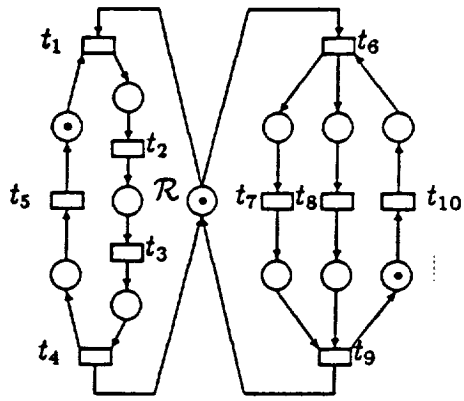
is a macrotransition reduction of $\mathcal{N}$ and θ is the macrotransition that replaces $\mathcal{N}'$. The system $\langle \mathcal{N}_r, M_r \rangle$ is called a macrotransition reduction of $\langle \mathcal{N}, M_0 \rangle$ where:

$$M_r = M_0 \mid P \setminus P' + K \cdot Post(t'_0)$$

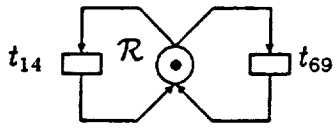
and K is the minimum number of tokens found in the different paths from t'_i to t'_0. In other words, M_r is the restriction of M_0 to the remaining places plus K tokens per output place of t'_0.

Figure 7.2 shows a macrotransition reduction. There exists one token in place c, thus the unique path from transition 3 to transition 1 has one token and $K = 1$.

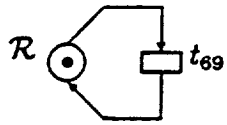
The value of the macrotransition concept lies in the following result that basically derives from MG systems theory:



(a) A shared resource is used by two different processes (one with internal concurrency)



(b) MT-reduction of the net system in (a)



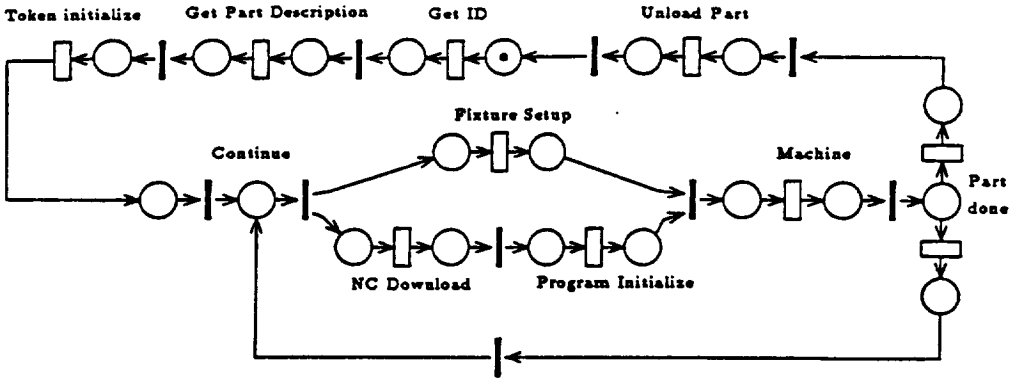
(c) An MP-reduction of the System in (b)

Figure 7.3: The Original Net is not Free-Choice, but MPMT (and simple)

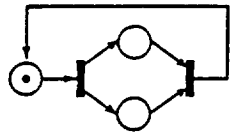
Property 7.2.2 *In the absence of an unmarked circuit (or p-semiflow) in $\mathcal{N}'$, the reduction of a macrotransition preserves liveness, boundedness (but not the bound of the system) and reversibility.*

Now the MPMT-nets can be recursively defined as follows:

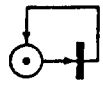
Definition 7.2.1 *A macroplace/macrotransition (MPMT) net is an ordinary net such that, disregarding the conditions on the marking in the MP- and MT-rules, it can be reduced to a simple self-loop of one place and one transition by recursively applying the MP and MT reduction rule (Definition 7.2.4 and 7.2.2).*



(a) Petri net representation of a controller [AJD90b]



(b) MP-reduction of the net system in (a)



(c) MT-reduction of the net system in (b)

Figure 7.4: An MPMT and FC net system

Figures 7.3 and 7.4 show the reduction (and refinement) processes leading to two different MPMT-nets. The reduction (refinement) also verifies the marking conditions on the MP- and MT-rules, therefore, both net systems are bounded, live and reversible.

Property 7.2.3 *The reverse and the dual (thus the reverse-dual also) of an MPMT-net is also an MPMT-net.*

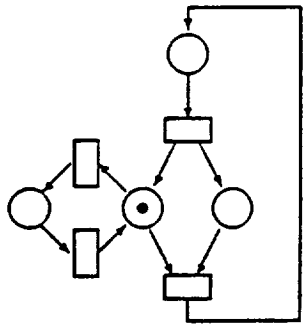


Figure 7.5: A Reversible Non-live MPMT-system

The net in Figure 7.3 is MPMT but not FC. The net in Figure 7.4 is both MPMT and FC.

How do we recognize if a given net is an MPMT-net? We need to find the subnets of a net which can then be reduced to a place or to a transition. For the reduction to a place, in [Sil81] an efficient (polynomial) algorithm for this purpose is given. It consists of first removing all transitions with more than one input or one output arc, which splits the net into one or more SMs. Then simple recursive procedures can be applied to each connected subnet to check conditions (b.1) and (b.2) from Definition 7.2.1. Now, for the reduction to a transition the same procedure is applied, but now first all places with more than one input or one output arc are removed. This splits the net into one or more MGs. Conditions (b.1) and (b.2) of Definition 7.2.3 can be checked by using simple procedures on each connected subnet. The entire process can be done in polynomial time.

The net in Figure 7.1 (Figure 7.2) is analyzed by applying first the reduction of subnets to a transition (place).

7.3 Basic qualitative properties of MPMT-nets

From the definitions it is not difficult to observe that MPMT-nets are strongly connected. For SMs, strong connectedness guarantees the existence of live markings (i.e., structural liveness) and for MGs the boundedness for any initial marking (i.e., structural boundedness). On the other side, SMs are conservative nets (i.e., $\exists Y > 0$ s.t. $Y^T \cdot C = 0$, thus, structurally bounded), while MGs are structurally live nets. Combining these facts, the following important structural properties hold.

Property 7.3.1 *MPMT-nets are:*

1. *Ordinary and strongly connected.*
2. *Conservative (thus, structurally bounded).*

3. *Structurally live (then also consistent, i.e., $\exists > 0$ s.t. $C \cdot X = 0$, because they are also conservative).*

The basic structural properties of MPMT-nets have already been presented. Behavioral properties can be studied in a straightforward way.

Property 7.3.2 *Let $\mathcal{N}$ be an MPMT-net. $\langle N, M_0 \rangle$ is a live net system iff all p-semiflows are marked:*

$$\forall Y \geq 0 \text{ s.t. } Y^T \cdot C = 0 \ , \ Y^T \cdot M_0 > 0$$

Corollary 7.3.1 *Liveness of MPMT net systems can be characterized in polynomial time: $\langle \mathcal{N}, M_0 \rangle$ is live iff there exists no unmarked P-semiflow. Analytically, $\langle \mathcal{N}, M_0 \rangle$ is live iff there exists no solution for the following linear system:*

$$\begin{aligned} & Y^T \cdot C = 0 \\ & Y \geq 0 \\ & Y^T \cdot M_0 = 0 \end{aligned} \tag{7.1}$$

Corollary 7.3.2 *Let $\mathcal{N}$ be an MPMT-net and $\mathcal{N}_r$ its reverse. $\langle \mathcal{N}, M_0 \rangle$ is live iff $\langle \mathcal{N}_r, M_0 \rangle$ is live.*

Property 7.3.3 *Let $\langle \mathcal{N}, M_0 \rangle$ be a live MPMT-system. The three following statements are equivalent:*

i) $M \in R(\mathcal{N}, M_0)$, i.e., M is reachable from M_0.

ii) $M = M_0 + C \cdot \vec{\sigma}$, with $M \in \mathbb{N}^n, \vec{\sigma} \in \mathbb{N}^m$.

iii) $B^T \cdot M = B^T \cdot M_0$, with B a basis of left annullers of the incidence matrix (i.e., $B^T \cdot C = 0$) and $M \in \mathbb{N}^n$

According to the above property, $M \in R(\mathcal{N}, M_0)$ iff $M_0 \in R(\mathcal{N}, M)$. In other words:

Corollary 7.3.3 *Live MPMT-systems are reversible, but the converse is not true. (see Figure 7.5).*

The bounds of places can be computed in polynomial time using Linear Programming Techniques [MJ88]:

Corollary 7.3.4 *Let $(\mathcal{N}, M)$ be a live MPMT-net system. The behavioral and structural bounds of place p coincide:*

$$B(p) = \max\{M(p) | M \in R(\mathcal{N}, M_0)\} =$$
$$SB(p) = \max\{M(p) | M = M_0 + C \cdot \vec{\sigma} \geq 0, \vec{\sigma} \geq 0\} \tag{7.2}$$

Property 7.3.3 relates directly to Theorem 3.4.3, while Corollary 7.3.4 is equivalent to Corollary 3.4.3 for MGs.

7.4 Well-formed SISO cuts

While in MGs, every SISO cut was a well-formed cut, in MPMT-nets we have to restrict the types of cuts which we can perform. Specifically we only cut the net at persistent places, i.e., places where $|p^\bullet| = 1$, i.e., once a token has entered an interface place, the routing out of it is deterministic.

Definition 7.4.1 (Well-formed SISO MPMT cut) *A cut ξ is well-formed if* $|q_{12}^\bullet| = |q_{21}^\bullet| = 1$.

With the above properties of structural liveness, reachability and the marking bound of places, we can apply the results of the previously introduced methods directly to MPMT-nets. Note that in general the vector of the visiting ratios can not be computed from the net structure.

As a final comment, we can now give a particular case when to use FEA in MPMT-nets. From Property 2.3.1, we know that FEA leads to exact aggregation

in PFQNs. Therefore, if we aggregate a macroplace (i.e., a subnetwork which is in product form) by flow equivalence, we will be able to replicate at least the MCT of the subnet. Nevertheless as we will see, the change in the coefficient of variation might still introduce considerable error in the aggregation.

7.5 Summary

In this chapter we have introduced a new subclass of Petri nets with extended modeling capabilities over SMs and MGs. Having established the structural similarity of MGs and MPMT-nets for the marking bounds and the reachability of markings, we can use any of the previously discussed approximation methods for SISO cuts in MPMT-nets without any modification for the iterative part. This will be demonstrated in the next chapter.

CHAPTER 8
Examples for SISO Cuts in MPMT-nets

8.1 Introduction

In this chapter, we present some numerical examples for SISO-cuts in MPMT-nets.

The first example considers the performance analysis of manufacturing transfer lines (or simply flow lines) which has received considerable attention in the literature due to their economic importance. In the first survey of this area [Koe59], the earliest papers date back 40 years, while today a vast amount of literature exists. In the most recent survey [DG91] 180 papers are cited which are directly related to their modeling and analysis. In the context of Petri nets, flow lines with reliable machines (FLRMs) can be modeled by marked graphs (MGs), while flow lines with unreliable machines (FLUMs) involve choice (i.e., the break down of machines) and therefore, can only be modeled using MPMT-nets.

An analytic solution for the very particular case of a short two machine, one buffer flow line with exponentially distributed machining, failure and repair time has been found [GB81]. In [CG87] this analytic solution of a short line has been used to devise an efficient approximation for longer flow lines whose exact analysis by solving the underlying CTMC becomes computationally intractable due to the problem of state explosion. Another approximation method is described in [DDL89] which employs an approximation based on continuous material flow. These approximation methods are problem specific, i.e., only suitable for the analysis of flow lines. For this example, we compare Response Time Approximation (RTA), Flow Equivalent Aggregation (FEA), Delay Equivalence (DE) and Marie's method.

In the second example, we examine an MPMT-net, whose structure is that

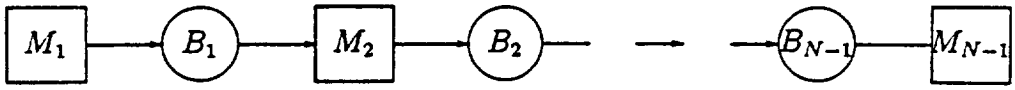


Figure 8.1: Flow Line with N Machines

of an uninterpreted dataflow graph [KBB87]. Here we use a hybrid approximation approach. In the first step a macroplace is reduced to a single transition by FEA. In the second step, the reduced net is analyzed by RTA and Marie's method and the accuracy is compared.

8.2 Modeling of Transfer Lines with Petri Nets

Figure 8.1 shows a flow line with N machines. Parts enter through M_1 and leave through M_N. We denote the i-th machine as M_i and the buffer which lies in between M_i and M_{i+1} as B_i. C_i is the capacity of B_i, i.e., the total number of parts which can be stored *in between* M_i and M_{i+1}. The current buffer content is denoted by c_i while $\overline{C}$ denotes the vector of the buffer capacities.

We now give a short review of flow line models and how they can be modeled using Petri nets. We concentrate on FLUMs with operation dependent failures, i.e., failures that can only occur when the machine is working. Time dependent failures, i.e., failures which do not depend on the machining operation and which can occur even if the machine is blocked or starved are not considered, but could also be incorporated into the Petri net model. Figure 8.2 introduces the notation for all of the transfer line models.

A common assumption of these models is that the first machine is never starved and the last machine is never blocked. The machining operation of M_i is represented by transition m_i with rate λ_i, failure by f_i with rate μ_i and the repair by r_i with rate θ_i. Note that the failure-repair model is a Macroplace. P_{B_i} represents B_i. If P_{M_i} is marked, M_i is currently idle (i.e., starved), whereas W_i is marked if the machine

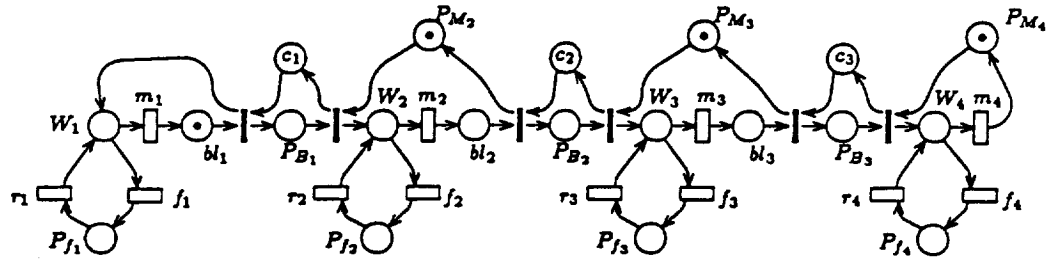


Figure 8.2: Blocking After Service (BAS) Flow Line

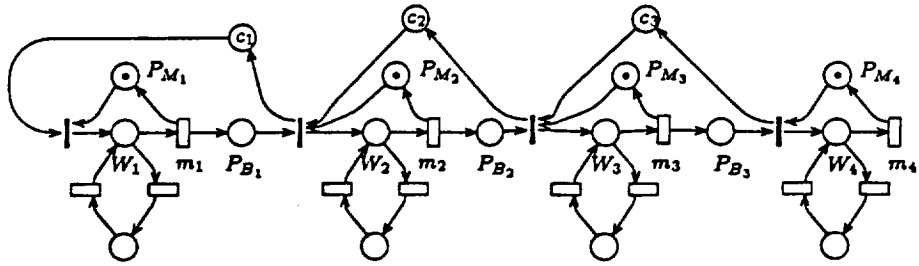


Figure 8.3: Blocking Before Service Parts not Occupied (BBS-PNO) Flow Line

is currently processing a piece. P_{f_i} is marked if the machine is down while bl_i is marked for a nonzero period of time if M_i is blocked. Where no confusion arises, the names of places and transitions are omitted from the Petri net models.

In [DG91], flow line models were divided into the following categories:

- Blocking After Service (BAS)(Figure 8.2):
  if at the instance of completion of a part at M_i the part cannot be put into the downstream buffer B_i because it is full, the machine is *blocked.* M_i will become operational only if a part is transported from B_i to M_{i+1} freeing one buffer space.

- Blocking Before Service, Parts Not Occupied (BBS-PNO) (Figure 8.3):
  here the part is not admitted into M_i if no buffer space is available in B_i. The BBS-PNO line with buffer capacity $\overline{C} + \mathbb{1}$ has the same throughput as a BAS line with capacity $\overline{C}$ ($\mathbb{1}$ is a vector of appropriate size whose entries

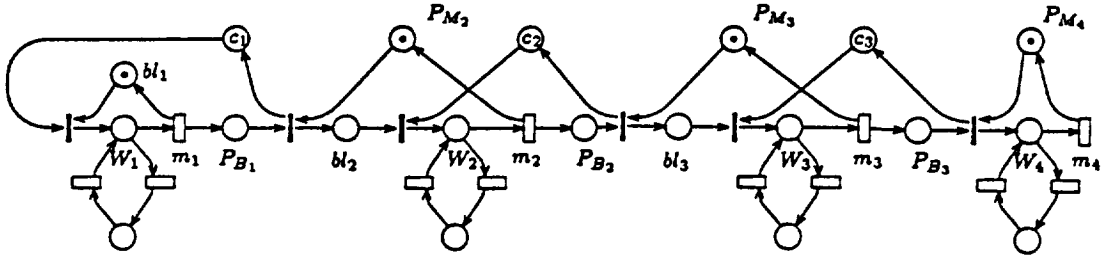


Figure 8.4: Blocking Before Service Parts Occupied (BBS-PO) Flow Line

are all one). This result was first established by Perros et al. [OP86] for the case of exponential timing, and has also been shown for the case of general distributions [DLT91]. Note that in BBS-PNO, if P_{M_i} is marked, M_i is starved of blocked.

- Blocking Before Service, Parts Occupied (BBS-PO) (Figure 8.4):
  In this model, the part is admitted into the machine, but operation does not start until a space in the output buffer is available.

Actually BAS is most likely to be encountered in manufacturing systems [DG91], e.g., in metal cutting operations there is no reason why a part should not be allowed into the system even if no output buffer space is immediately available.

Flow lines with reliable machines can be constructed by simply deleting the failure circuit, i.e, f_i, r_i and P_{f_i} from all machines.

8.2.1 Choong and Gershwin Approximation in Flow Lines

The basic idea is to decompose an N machine transfer line into $N-1$ modules [CG87] consisting of the buffer B_j, $M_u(j)$ (the upstream machine) and $M_d(j)$ (the downstream machine), which can analytically be solved by the method outlined in [GB81]. In Figure 8.5, the five machine flow line is decomposed into four modules. Each of the modules is characterized by six parameters, three for each machine: λ_u, μ_u, θ_u and λ_d, μ_d, θ_d. The goal of the approximation is to find these 6 parameters such that the pseudo-machines capture the *upstream* and *downstream* behavior of

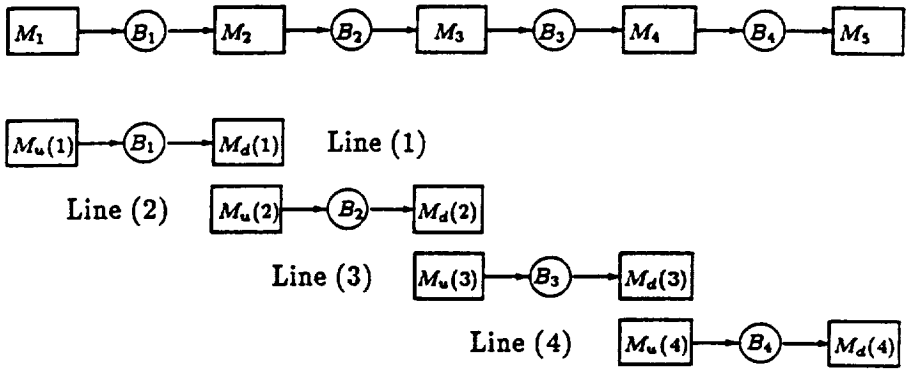


Figure 8.5: Decomposition Method of Choong and Gershwin

the flow line, i.e., the rates with which parts are processed if the line upstream is up (λ_u^j), the rate with which the upstream part of the line goes from the up to down state (μ_u^j) and vice versa (θ_u^j). A similar discussion holds for λ_d^j, μ_d^j and θ_d^j.

The stand-alone efficiency of a machine is given by:

$$\rho_i = \frac{\lambda_i \theta_i}{\theta_i + \mu_i} \tag{8.1}$$

In the line, the production rate will, in general, be smaller because there is a nonzero probability that the machine is blocked or starved (or both). The efficiency E of the machine in the line is therefore given by

$$E_i = \rho_i \text{ prob}[c_{i-1} > 0 \text{ and } (c_i < C_i \text{ and } M_i \text{ not blocked})] \tag{8.2}$$

The conservation of flow implies that

$$E_1 = E_2 = \ldots E_{N-1} = E_N \tag{8.3}$$

If the probability of the event "machine blocked and starved at the same time is small, equation (8.2) can be simplified to

$$E_i = \rho_i(1 - \text{ prob}[c_{i-1} > 0] - \text{prob}[c_i < C_i \text{ and } M_i \text{ not blocked}]) \tag{8.4}$$

The next step is to examine λ, μ and θ in the modules and how they relate to the machine parameters. From the examination of the interruption and resumption of flow at the modules and together with equations (8.3) and (8.4) a set of six equations per module can be derived, which then have to be solved iteratively. The method breaks down if the line is strongly unbalanced and $\rho_i \gg \rho_j$.

8.2.2 Experimental Results: Three Machine Transfer Line

The flow lines which are analyzed in this section were previously described and analyzed in [CG87]. The lines are of the type BBS-PNO, nevertheless, here the flow lines were modeled as BAS with the appropriate change in the buffer size as outlined in Section 8.2.

As the convergence criterion, we chose that the throughputs obtained during the iteration do not change by more than 0.1%.

In order to interpret the results, it is useful to look at the Moment Generating Function of the time x it takes to move a token from place W_i to bl_i. The moment generating function, $T(s)$ can be derived by using the techniques outlined in [GDZ90]. First we transform the macroplace into a construct where the conflict is given by immediate transitions according to Theorem 3.5.1 and is presented in Figure 8.6.

The transformed macroplace forms a *loop structure*. $W_1(s)$ represent the loop, while $W_2(s)$ represents the "gain" after the loop as shown in Figure 8.7. First we find the stochastic "transfer function" $W_1(s)$ of the loop given by $\{f^i, f^t, r\}$, i.e., two exponentials in series times the probability that path is taken.

$$W_1(s) = \frac{\mu\theta}{(s-(\mu+\lambda))(s-\theta)} \tag{8.5}$$

$W_2(s)$ is the "gain" after the loop and is simply an exponential times the probability

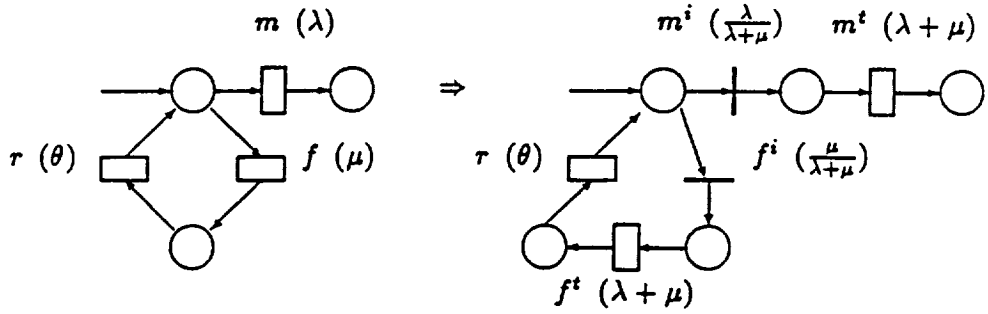


Figure 8.6: Transformation of Macroplace in Unreliable Machines (Probabilities and Rates of Transitions in Parentheses)

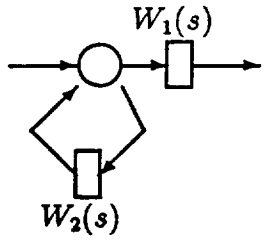


Figure 8.7: Loop Structure for Finding Moment Generating Function

the path is taken.

$$W_2(s) = \frac{\mu}{s - (\mu + \lambda)} \tag{8.6}$$

The overall transfer function is given by

$$T(s) = \frac{W_2(s)}{1 - W_1(s)} \tag{8.7}$$

By substituting equation (8.5) and (8.6) into (8.7), the moment generating function is obtained:

$$T(s) = \frac{\lambda(\theta - s)}{s^2 - s(\lambda + \mu + \theta) + \lambda\theta} \tag{8.8}$$

By differentiating the $T(s)$ with respect to s, it is now easy to determine the first two moments and obtain the mean and variance:

$$E[x] = \frac{1}{\rho} = \frac{\theta\mu}{\theta + \mu} \qquad E[x^2] = 2\frac{\lambda\mu + (\theta + \mu)^2}{\theta^2\lambda^2} \tag{8.9}$$

The coefficient of variation (CV) is a measure for the dispersity of the probability density function (pdf) (for the case of an exponential pdf $CV = 1.0$). For the model at hand, the CV is given by:

$$CV[x] = \sqrt{\frac{E[x^2] - E[x]^2}{E[x]^2}} = \sqrt{1 + \frac{2\mu\lambda}{(\theta + \mu)^2}} \tag{8.10}$$

Note that with this information it is possible to transform a transfer line with unreliable machines to a model with reliable machines by substituting the machining rate in the FLRMs by the pdf obtained from eq. (8.8) [DG91]. This corresponds to finding the *flow equivalent* from MP_i to bl_i. By doing so, we disregard the higher moments of the pdf, by substituting an exponential where $CV = 1.0$.

The parameters of the three machine line are shown in Table 8.1. This short flow line is analytically tractable (the underlying CTMC has 798 tangible states). Rather than comparing our results to the simulation published in [CG87], we chose to compare our approximation results for the short line to the exact solution of the CTMC (which by and large confirms the simulation).

The results of the approximation are shown in Table 8.2. Cut ξ_A traced through P_{M_i} and bl_i introduces the single level decomposition as shown in Figure 8.8 (here only the cut, but not the aggregated net systems are shown). The first four columns show the throughput obtained with the exact solution of the underlying CTMC, together with the mean and coefficient of variation of the effective machining time of machine 2. Columns 5 and 6 show the number of iterations as well as the error introduced by using RTA (remember that because of Property 5.4.2, all presented iterative methods yield the same result for $K = 1$). Columns 8 through 11 show the results using a hierarchical decomposition approach as shown in Figure 8.8. We apply the first cut (ξ_1) at buffer 1 which results in $\mathcal{AS}_1^1$, $\mathcal{AS}_2^1$ and $\mathcal{BS}^1$. At the next level, we apply cut ξ_2 at buffer 2 in $\mathcal{AS}_2^2$ (now at level 1) to yield the aggregated net systems $\mathcal{AS}_{21}^2$, $\mathcal{AS}_{21}^2$ and $\mathcal{BS}^2$. Figure 8.8b shows the resulting tree with the basic skeletons.

i	Machine Parameters					C
	θ_i	μ_i	λ_i	ρ_i	CV_i	
1	0.05	0.03	0.5	0.3125	2.38	8
2	0.06	0.04	–	–	–	8
3	0.05	0.03	0.5	0.3125	2.38	

Table 8.1: Parameters for 3 Machine Transfer Line

Column 12 shows the results obtained by using FEA at the macroplaces representing the machine failures and repairs. In the three machines, W_1, m_i, f_i, P_{f_i} and r_i are substituted by a single transition with the stand alone machining rate ρ_i, which is computed with equation 8.9. Finally, column 13 shows the results obtained using the decomposition approach by Choong.

For the SISO cut ξ_A, $\mathcal{AS}_1$ has 59 tangible states ($\mathcal{AS}_2$ has 21). In the hierarchical decomposition, $\mathcal{AS}_1^1$ and $\mathcal{AS}_{22}^2$ have 19 while $\mathcal{AS}_{21}^2$ has 180 tangible states.

8.2.2.1 Interpretation of Results

All approximation methods for the SISO cut at machine 2 yields a bad approximation of the throughput. Three factors contribute:

1. The number of tokens at that cut is low ($K = 1$).

2. In [ABS84], Aggrawal et al. observed that response time preservation is an effective technique for low to moderate coefficients of variation ($CV < 5$) and a large number of customers ($K > 5$). From Table 8.2 we can see that the error increases rapidly as λ_2 (and therefore CV_2) increases.

3. An additional error is introduced by cutting the *fastest machine.*

The results in Table 8.2 derive from an MPMT-net. In order to examine the source of the rather large error, let us now analyze a similar marked graph system, i.e., let us analyze a transfer line without failures (a marked graph!) whose

All Errors in %												
Analytical Results				SISO cut ξ_A			Hierarchical				FEA	Choong
				RTA		FEA	RTA		Marie		MP	
λ_2	ρ_2	CV_2	$\mathcal{X}$	r	Err	Err	r	Err	r	Err	Err	Err
0.1	0.06	1.3	.0598	2	0.2	-0.1	2	-0.2	4	0.0	0.1	0.1
0.2	0.12	1.6	.1154	3	0.7	0.8	3	-2.5	4	0.0	1.8	-0.2
0.3	0.18	1.8	.1592	3	2.0	3.2	3	-5.7	5	-0.4	7.9	-0.1
0.4	0.24	2.0	.1897	4	3.7	6.0	3	-8.2	7	-0.8	17.0	-0.3
0.5	0.30	2.2	.2093	4	5.9	8.0	3	-9.4	7	-1.1	23.4	-0.1
0.6	0.36	2.4	.2216	5	8.0	9.0	3	-9.7	7	-1.3	25.0	0.5
0.7	0.42	2.6	.2296	5	9.8	9.4	3	-9.7	9	0.8	24.3	0.5
0.8	0.48	2.7	.2350	5	11.4	9.4	3	-9.4	9	0.2	23.1	1.2
0.9	0.54	2.9	.2388	6	12.6	9.3	3	-9.1	9	0.0	22.0	1.6
1.0	0.60	3.0	.2416	6	13.5	9.2	3	-8.7	11	0.7	21.1	1.7
1.2	0.72	3.3	.2454	7	14.9	8.9	3	-8.0	16	1.1	19.9	2.6
1.5	0.90	3.6	.2489	8	16.1	8.6	3	-7.3	19	0.0	18.6	3.1
1.6	0.96	3.7	.2497	8	16.5	8.5	3	-7.0	–	–	18.3	3.2
2.5	1.50	4.6	.2538	10	17.8	7.9	3	-5.7	–	–	16.8	4.0
3.0	1.80	5.0	.2549	11	18.1	7.8	2	-5.4	–	–	16.4	4.4

Table 8.2: Results for Three Machine Transfer Line

machining rates are the stand alone machining rates ρ_i (equation 8.9) of the three machine, two buffer transfer line. Again, the SISO cut is performed at P_{M_2} and bl_2. Table 8.3 shows the results (ρ_2 is the effective machining rate of machine 2, this is the value which was used in the marked graph system).

From these results, we can draw two conclusions:

- The aggregation in the marked graph system leads to a reasonable accuracy, i.e., the maximum error is only 5.1%. From a stochastic perspective, the difference between the system corresponding to Table 8.2 and Table 8.3 is a change in the coefficient of variation of the machining time. In the MPMT-net (Table 8.2), the coefficient of variation is larger than one, while in the marked graph system (Table 8.3), all coefficients of variation are equal to one.

Analytical			SISO Cut ξ_A	
λ_2	ρ_2	χ_{exact}	χ_{ξ_A}	Err
0.1	0.06	.0600	.0600	-
0.2	0.12	.1200	.1200	-
0.3	0.18	.1794	.1794	-
0.4	0.24	.2331	.2327	.1%
0.5	0.30	.2689	.2690	-
0.6	0.36	.2853	.2887	1.2%
0.7	0.42	.2916	.2986	2.4%
0.8	0.48	.2940	.3038	3.3%
0.9	0.54	.2950	.3067	4.0%
1.0	0.60	.2955	.3084	4.4%
1.2	0.72	.2961	.3101	4.7%
1.5	0.90	.2964	.3111	5.0%
1.6	0.96	.2964	.3112	5.0%
2.5	1.50	.2967	.3118	5.1%
3.0	1.80	.2967	.3119	5.1%

Table 8.3: Results for Three Machine Transfer Line Modeled with Reliable Machines

Therefore, we conclude that the poor accuracy of the SISO cut through the MPMT-net is due to the large coefficients of variation of the equivalent machining times.

- From Table 8.3, it can be seen that as soon as λ_2 increases beyond 0.52 (where the flow line is approximately balanced, i.e., $\rho_1 = \rho_2 = \rho_3 = 0.3125$) the error increases rapidly. Therefore, we conclude that a cut at the fastest machine can lead to a larger error.

The hierarchical decomposition using RTA also yields a substantial error. The problem with Marie's method however, is that the number of required iterations is larger than RTA and that it fails to converge for large values of λ_2 ($\lambda_2 > 1.6$). The iteration simply cycles and does not converge.

Aggregating the macroplaces yields a poor result (column 12 in Table 8.2). All we did here was to reduce the coefficient of variation to unity. As we reduced a

	8 Machine Parameters						7 Machine Parameters					
i	θ_i	μ_i	λ_i	ρ_i	CV_i	C_i	θ_i	μ_i	λ_i	ρ_i	CV_i	C_i
1	0.3	0.03	1.5	1.3636	1.35	4	0.3	0.02	0.20	0.1875	1.04	2
2	0.5	0.05	1.3	1.1818	1.20	3	0.4	0.05	0.23	0.2044	1.06	2
3	0.1	0.01	2.0	1.8182	2.08	4	0.1	0.01	0.30	0.2727	1.22	4
4	0.4	0.06	1.6	1.3914	1.38	2	0.4	0.07	0.26	0.2213	1.08	2
5	0.3	0.04	2.0	1.7648	1.54	4	0.3	0.03	0.21	0.1909	1.06	2
6	0.1	0.01	1.7	1.5455	1.95	2	0.1	0.03	0.27	0.2077	1.40	4
7	0.3	0.02	1.2	1.1250	1.21	5	0.4	0.06	0.26	0.2261	1.07	-
8	0.4	0.05	1.6	1.4222	1.34	-	-	-	-	-	-	-

Table 8.4: Parameters for Long Flow Lines

macroplace, the MCT of the subnet is independent of the interarrival process. Here we see clearly that the dependence of the MCT on the interarrival time is not the only factor which introduces an error in the aggregation, but that the coefficient of variation can have a strong impact on the throughput.

For the SISO cut ξ_A, FEA yields a better approximation than the presented iterative methods. This is coincidental and due to the interplay of the change of the coefficient of variation of the interface transition and the dependence of the MCT on the interarrival time of tokens. This relatively good results is specific to this example, as we will see in the next section.

8.2.3 Experimental Results: Long Transfer Lines

Table 8.4 shows the machine parameters for the 7 and 8 machine line. Both lines were analyzed by SISO cuts and hierarchical decomposition. Table 8.5 and 8.6 show the numerical results of the approximation. Finally Table 8.7 shows a comparison of the tangible state space for the SISO cuts.

The hierarchical decomposition of the subnets (Figure 8.9) yields a balanced binary tree. For the ease of notation, the aggregated net system containing M_i is

All Errors in %, DE = Delay Equivalence Throughput using Simulation: $\mathcal{X} = 0.1333$										
Choong	SISO M_4			SISO B_3, $K = 4$						
	RTA		FEA	RTA		Marie		DE		FEA
Err	r	Err	Err	r	Err	r	Err	r	Err	Err
-2.2	4	2.9	-44.6	2	1.8	4	0.6	20	3.0	-13.5
Hier M		Hier. Buff								
RTA		RTA		Marie						
r	Err	r	Err	r	Err					
5	6.6	5	9.8	10	2.2					

Table 8.5: Results for 7 Machine Flow Lines (Number of Iterations are Given at Top Level)

All Errors in % Throughput using Simulation: $\mathcal{X} = 0.8304$								
Choong	SISO M_5			SISO B_4, $K = 2$				
	RTA		FEA	RTA		Marie		FEA
Err	r	Err	Err	r	Err	r	Err	Err
0.0	6	7.4	-46.5	5	6.6	3	3.6	-25.0
Hier M		Hier. Buff						
RTA		RTA		Marie				
r	Err	r	Err	r	Err			
11	0.4	4	0.2	9	7.1			

Table 8.6: Results for 8 Machine Flow Line (Number of Iterations are Given at Top Level)

State Space of ($\mathcal{AS}_i$ at SISO cuts							
7 Mach, M_4		7 Mach, B_3		8 Mach, B_4		8 Mach, M_5	
$\mathcal{AS}_1$	$\mathcal{AS}_2$	$\mathcal{AS}_1$	$\mathcal{AS}_2$	$\mathcal{AS}_1$	$\mathcal{AS}_2$	$\mathcal{AS}_1$	$\mathcal{AS}_2$
2181	843	685	8133	9115	9793	32211	1429

Table 8.7: Tangible State Space for SISO Cuts at Long Flow Lines

denoted by $\mathcal{AM}_i$. Obviously, the individual aggregated net systems for the hierarchical decomposition of the long lines have the same structure as in the three machine case.

For the SISO cuts at the 7 machine line the quality of the approximation using an iterative method (here $K = 1$) is quite good. The cut at B_3 is superior to the one at M_4, due to the larger number of tokens in the circuit that is cut. Note, that here the coefficient of variation of almost all machines is close to unity.

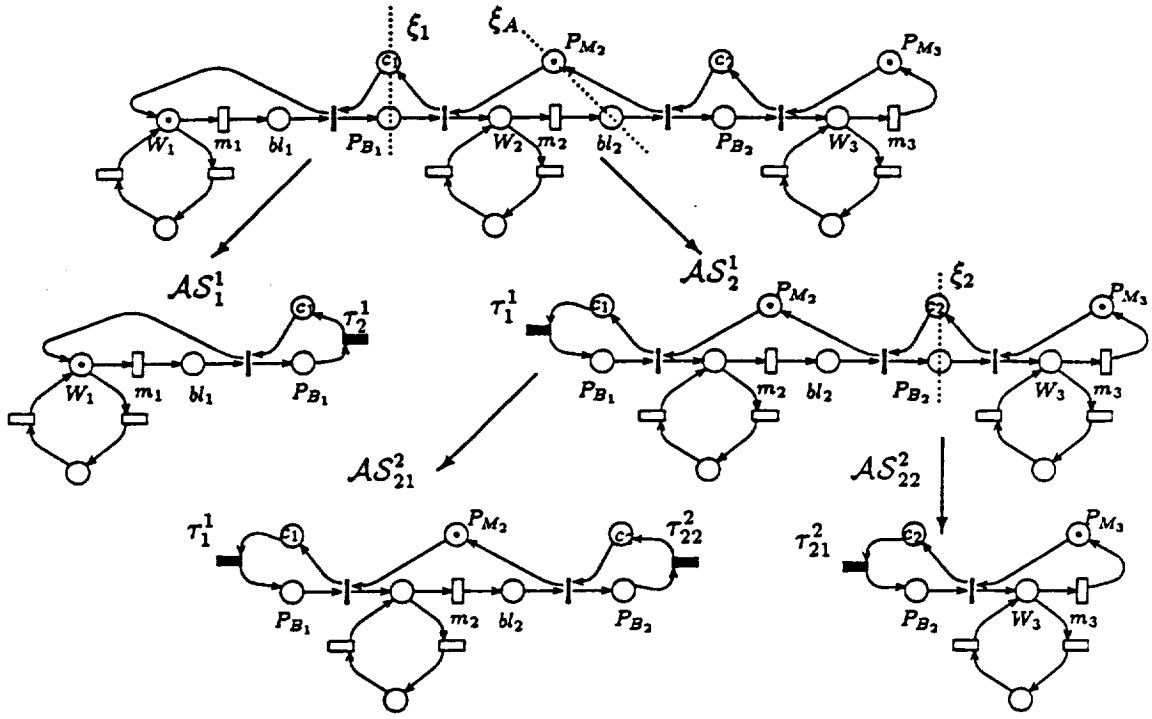
For the SISO cuts at the 8 machine flow line, the coefficient of variations of the effective machining times are comparatively large which accounts for the error at the SISO cut at M_5. The cut at B_4, is superior to the cut at M_5, because now the number of customers is larger ($C_4 = 2$) than in the cut at M_5. Again, Marie's method yields the best SISO approximation while FEA is poor. In the 8 machine example we did not use delay equivalence because of the high computational cost of analyzing the underlying CTMC and the slow convergence of the method.

The results of the hierarchical decomposition are acceptable. Nevertheless from just looking at the transfer line, it is not possible to infer which cuts (machines or buffer) and which method (RTA or Marie's method) yields the best approximation.

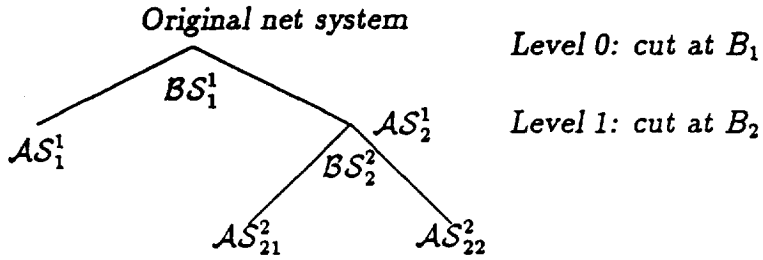
From the above results it becomes clear that *single cuts should be performed at the buffers*, because the number of customers (tokens) is usually higher than at the machines. RTA often yields the fastest convergence, while Marie's method yields the best results, although it sometimes fails to converge.

8.3 Uninterpreted Dataflow Graph

Figure 8.10 shows the *structure* of an uninterpreted dataflow as given in [KBB87]. The net is MPMT and free-choice. For a physical interpretation of the model, the marking has to be safe, nevertheless we analyze the net with the shown



(a) Cuts and Aggregated Net Systems



(b) Binary Tree

Figure 8.8: 3 Machine, 2 Buffer Hierarchical Decomposition

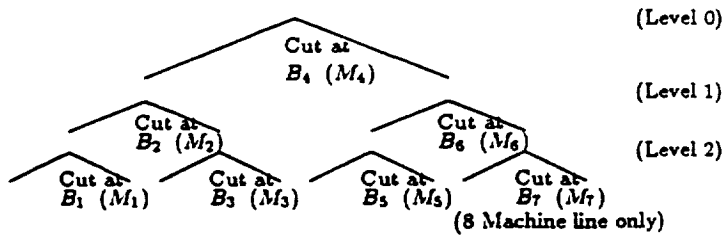


Figure 8.9: Binary Tree for Aggregated Net Systems for Long Flow Lines

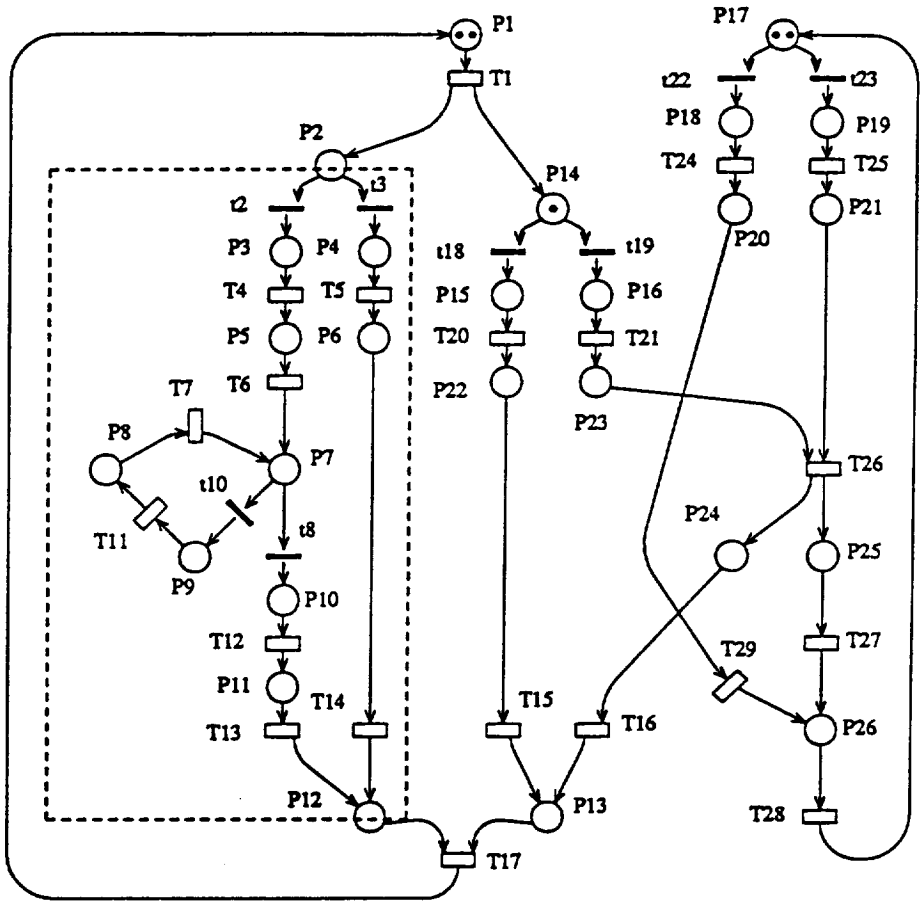


Figure 8.10: MPMT-net with the Structure of Uninterpreted Dataflow Graph

(non-safe) initial marking. The interfiring times of the transitions are as follows:

$$\begin{aligned} s_1 &= s_4 = s_5 = s_{24} = s_{25} = 1.0 \\ s_6 &= s_7 = s_{20} = s_{21} = s_{26} = 2.0 \\ s_{11} &= s_{12} = s_{13} = s_{14} = s_{15} = s_{16} = 3.0 \\ s_{17} &= s_{27} = s_{28} = s_{29} = 4.0 \end{aligned} \tag{8.11}$$

The probability switches at the immediate transitions is given by the following probability rates:

$$t_2 = 0.333333,\ t_3 = 0.666667,\ t_8 = 0.857142,\ t_{10} = 0.142858 \tag{8.12}$$

$$t_{18} = 0.25, \quad t_{19} = 0.75, \quad t_{22} = 0.50, \quad t_{23} = 0.50$$

The vector of visit ratios can be computed according to equation (3.17), i.e., once we know the throughput of any transition, we can compute the throughput of all other transitions.

The visit ratio normalized for the throughput of T_1 is given below:

$$\vec{v} = \begin{bmatrix} T_1 \ (1.0000) \\ T_4 \ (0.3333) \\ T_5 \ (0.6667) \\ T_6 \ (0.3333) \\ T_7 \ (0.0556) \\ T_{11} \ (0.0556) \\ T_{12} \ (0.3333) \\ T_{13} \ (0.3333) \\ T_{14} \ (0.6667) \\ T_{15} \ (0.2500) \\ T_{16} \ (0.7500) \\ T_{17} \ (1.0000) \\ T_{20} \ (0.2500) \\ T_{21} \ (0.7500) \\ T_{24} \ (0.7500) \\ T_{25} \ (0.7500) \\ T_{26} \ (0.7500) \\ T_{27} \ (0.7500) \\ T_{28} \ (1.5000) \\ T_{29} \ (0.7500) \end{bmatrix} \tag{8.13}$$

In the first step we apply flow equivalent aggregation at the subnet shown in the dashed box in Figure 8.10. The subnet is a macroplace, i.e., here the mean

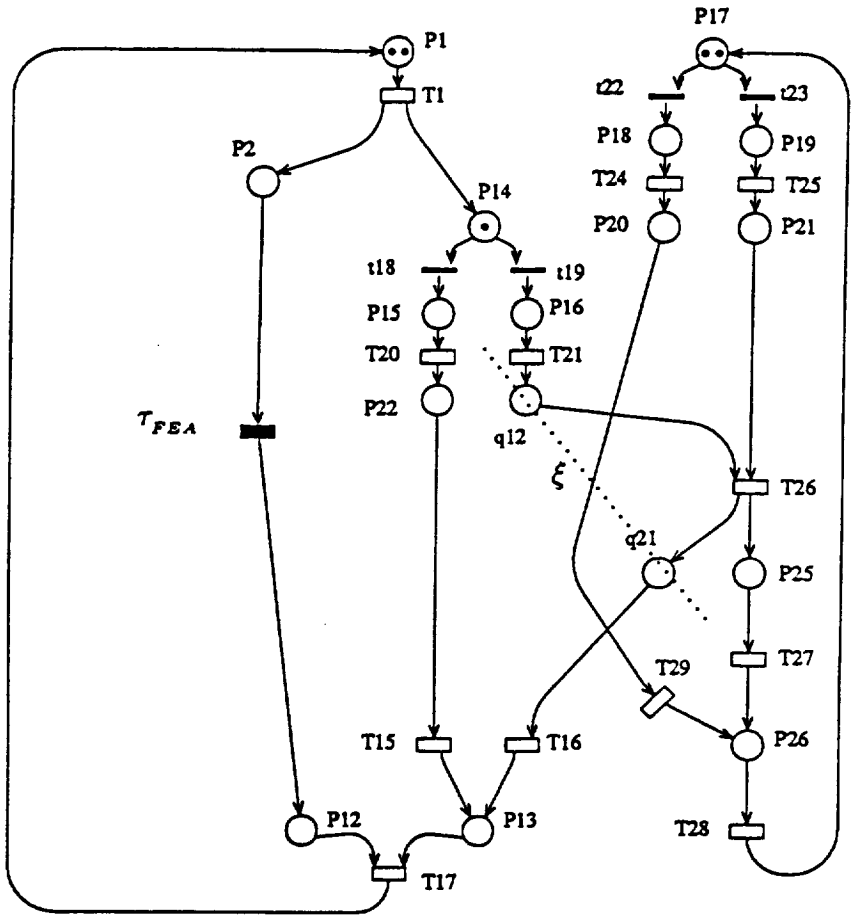


Figure 8.11: Uninterpreted Dataflow Graph with Flow Equivalent Transition

completion time of a token does not depend on the interarrival process. Table 8.8 shows the state dependent service rates of τ_{FEA} in Figure 8.11.

In the second step, the resulting net system shown in Figure 8.11 is further partitioned by ξ to form SN_1 to the left of the cut and SN_2 to the right of the cut. By following the SISO Decomposition rule (Section 4.3), we arrive at the aggregated net systems in Figure 8.12. The nets are already shown with a solution to LPP3, here $K = 3$.

Table 8.9 shows the results of the approximation using Marie's method and RTA. For both methods, the introduced error is under 2.0%. Again, Marie's method

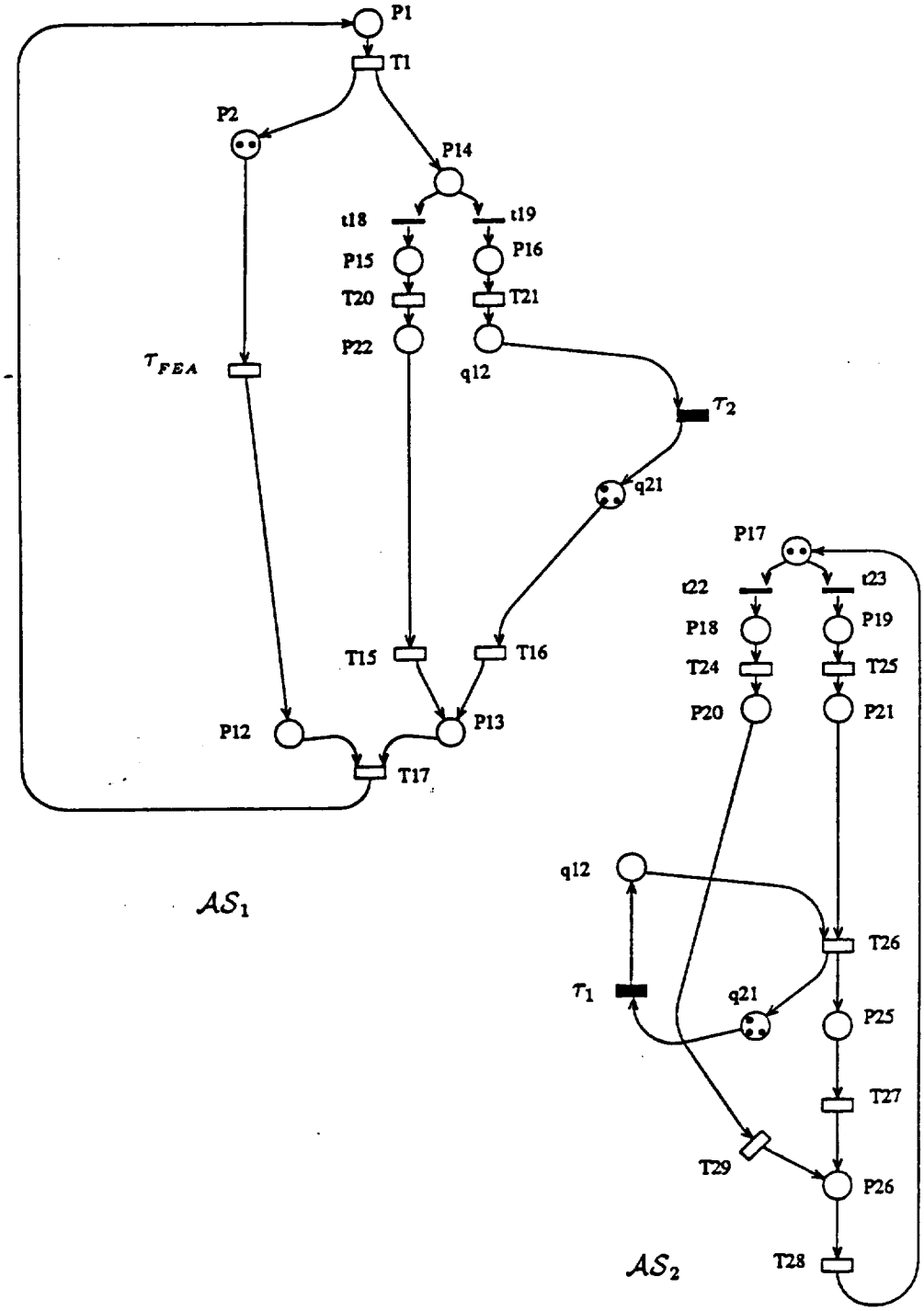


Figure 8.12: Aggregated Net Systems for Uninterpreted Dataflow Graph

	$k=1$	$k=2$
conditional throughput	0.168224	0.280533
tangible state space	8	36

Table 8.8: State Dependent Service Rates for τ_{FEA} in Dataflow Graph

$\mathcal{X}_{\text{exact}}(T_1) = 0.100859$			
RTA		Marie	
r	Err	r	Err
3	-1.75%	4	-0.76%

Table 8.9: Iteration Results for Uninterpreted Dataflow Graph (Error in %)

shows improved accuracy at the expense of a higher number of iterations. The original net has 57015 states, while $\mathcal{AS}_1$ has 261 states and $\mathcal{AS}_2$ has 84 states. Therefore, the reduction of the state space is by over 2 orders of magnitude ($57015/261 \approx 264$).

8.4 Summary

The results indicate that SISO cuts show good accuracy whenever the number of tokens in the cut circuit is high. Potentially bad results can occur if $K = 1$. For SISO cuts, in general Marie's method shows improved accuracy over RTA at the expense of a higher number of iterations. Sometimes, Marie's method fails to converge. Delay equivalence converges very slowly and does not show improved accuracy over RTA or Marie's method.

As can be seen from the examples, FEA can lead to potentially bad results because of two reasons:

1. If the MCT depends on the interarrival process, the interaction between subsystems is ignored. This error occurs when cutting transfer lines at the machines.

2. The coefficient of variation of the service times of transitions strongly influences the throughput (as could be seen at the 3 machine transfer line with reliable machines example). But even the iterative method could not have coped with that problem. For the particular case considered the results using FEA or RTA would have been the same (Property 5.4.2).

Unfortunately, none of the presented methods is consistently better than the Choong and Gershwin method for the solution of transfer lines. Nevertheless, with the above discussion it should be clear that SISO cuts lead to reasonable accuracy if the number of tokens in the cut circuit is high.

The hierarchical decomposition shows varying degrees of accuracy and at present it is not clear when it is advantageous to use SISO cuts or hierarchical decomposition.

CHAPTER 9

Flow Equivalent Aggregation for SIMO-cuts in MPMT-nets

9.1 Introduction

In Chapter 7 we have introduced SISO cuts for MPMT-nets. In this chapter we generalize SISO cuts for flow equivalent aggregation to SIMO cuts. The basic idea is to reduce a SIMO subnet to a state machine. We then formulate a reduced net called the *flow equivalent net* (FEN) [JD91a, JD91b]. The FEN replicates the input/output behavior of the SIMO subnet, as well as the expected delay a customer suffers.

9.2 Well-Formed SIMO MPMT Cut

Let us first formalize a *well formed* SIMO cut. Consider the net system with the cut ξ as shown in Figure 9.1.

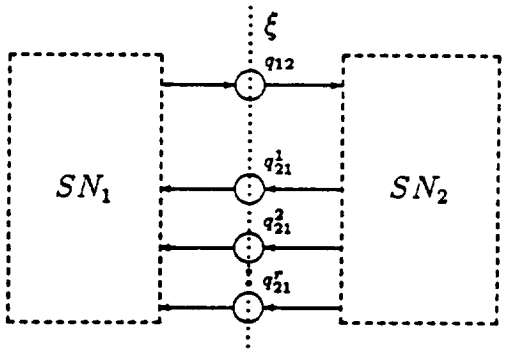


Figure 9.1: MPMT-net with SIMO Cut ξ and Interface Places

Definition 9.2.1 (Well-Formed SIMO MPMT Cut) *Let SN_1 and SN_2 be the subnets generated by a SIMO cut ξ to a MPMT-net, q_{12} the single (input) interface*

and $Q_{21} = \{q_{21}^1, q_{21}^2 \cdots q_{21}^r\}$ *with* $r = |Q_{21}|$ *the multiple (output) interface.* ξ *is* well-formed *iff:*

1. $|q_{12}^\bullet| = 1$ *and* $\forall q \in Q_{21}, |q^\bullet| = 1$.
2. *By the recursive reduction of macrotransitions and macroplaces,* SN_2 *can be reduced to a state machine.*

Definition 9.2.1 (1) states that once a token has entered an interface place, its flow is deterministic (the interface place ha only a single output transition), while (2) essentially states that we are not allowed to cut macrotransitions. Note that the interface of a well formed SIMO cut satisfies the *Product Form Decomposition* of Definition 2.3.1, i.e., once a token enters an interface, its routing is deterministic and we are not allowed to cut at concurrent processes (macrotransitions).

SN_2 can be structurally reduced to a state machine. Therefore, for the FEN we can also choose any state machine representation, as long as there exists a directed path from the input interface q_{12} to any of the output interface places in Q_{21}, and we do not violate condition (1) of Definition 9.2.1. Of course, we are interested in a *minimal representation* in order to reduce the state space as much as possible. Figure 9.2 shows two realizations.

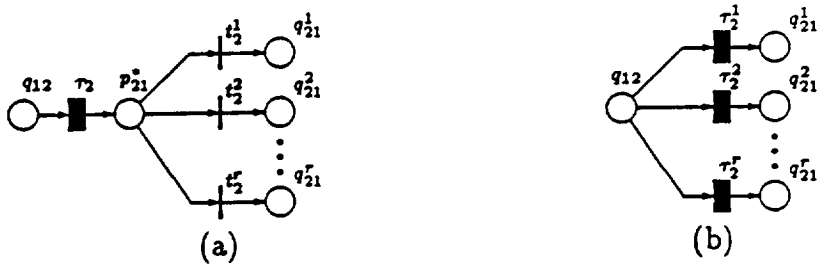


Figure 9.2: Two Equivalent Minimal Realizations for a SIMO Flow Equivalent Net

According to Theorem 3.5.1, both representations are equivalent. Note that the realization in Figure 9.2.a introduces additional vanishing markings because p_{21}^* is never marked for any nonzero period of time. The representation in Figure 9.2.a

directly shows the delay of the token (via the firing rate of τ_2) and the routing probabilities (via the immediate transitions $\{t_{21}^1 \ldots t_{21}^r\}$). As we are interested in a maximum reduction of the state space, we will use the realization shown in Figure 9.2.b which does not introduce vanishing markings.

Definition 9.2.2 (SIMO-MPMT Flow Equivalent Net) *Let $\mathcal{N} = (P, T, F)$ be an MPMT-net and ξ a well formed SIMO-MPMT cut through the interface places q_{12} and Q_{21} defining two subnets $SN_i = (P_i, T_i, F_i)$ with $i = 1, 2$, where $Q = \{q_{12} \cup Q_{21}\}$ with $|Q_{21}| = r$ as shown in Figure 9.1. The structure of the FEN is given by:*

$$\begin{aligned} q_{12}^{\bullet} &= \{\tau_2^1, \tau_2^2 \ldots \tau_2^r\} = \tilde{\tau_2} \\ \tau_2^{i\bullet} &= q_{21}^i, \quad i = 1 \ldots r \end{aligned} \tag{9.1}$$

as shown in Figure 9.2.

The goal of the quantitative analysis is to find the state dependent firing rates of τ_2^i, $\mu_2^i(n)$, $i = 1 \ldots r$, $n = 1 \ldots K$, where K is the maximum number of customers in SN_2.

By reducing SN_2 to the FEN, the original net system $\langle \mathcal{N}, M_0 \rangle$ leads to the flow aggregated net system, $\mathcal{FS} = \langle \mathcal{FN}, M_0 \rangle$ as shown in Figure 9.3. Structurally speaking from $\mathcal{N}$, SN_2 is reduced to the FEN (Definition 9.2.2). The structural reduction process is formalized by the following rule.

<u>Rule 3: SIMO-FEA Reduction Rule</u>

1. Structure: $\mathcal{FN} = (\tilde{P}, \tilde{T}, \tilde{F})$, is a flow aggregated net, where:

   - $\tilde{P} = P_1$
   - $\tilde{T} = T_1 \cup \tilde{\tau_2}$

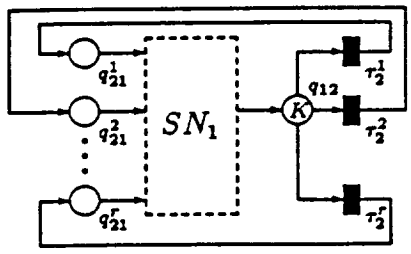


Figure 9.3: SN_1 with Substituted Flow Equivalent Net Forming $\mathcal{FS}$

- $\tilde{F} = F_1 \cup (\{q_{12}\} \times \tilde{\tau_2}) \cup (\tilde{\tau_2} \times Q_{21})$; $[\Rightarrow q_{12} =^\bullet \tilde{\tau_2}, Q_{21} = \tilde{\tau_2}^\bullet]$

2. **Marking:**

   Let $\overline{M}$ be a solution to the following linear programming problem:

$$\begin{aligned} &\textit{maximize} && \overline{M}(q_{12}) \\ &\textit{subject to} && \overline{M} = M_0 + C \cdot \vec{\sigma} \geq 0 \\ & && \vec{\sigma} \geq 0, \ \overline{M} \geq 0 \end{aligned} \qquad \text{(LPP6)}$$

   where C is the incidence matrix of the canonical MPMT-net obtained from the original net and cut. The initial marking for the flow aggregated net is the projection of $\overline{M}$ on $\mathcal{FN}$ (Definition 3.3.2 in Section 3.3).

The maximum number of customers K in SN_2 is given by the marking bound at the single interface place $K = B(q_{12}) = \overline{M}(q_{12})$.

Property 9.2.1 (SIMO-MPMT Reduction Rule) *Let $\langle \mathcal{N}, \mathcal{M}_0 \rangle$ be a live MPMT-net.*

1. *For the reduction from the original net system to the flow aggregated net system, the bounds of the places are preserved:*

$$B(p) = \tilde{B}(p), \ \forall p \in P_i$$

2. *Liveness in the flow aggregated net system is preserved.*

Proof: It is a direct consequence of the reduction of the SIMO subnet by recursive application of the MP and MT rules, as well as Corollary 7.3.4. The transformation of the marking empties the least-loaded paths from the input interface to the output interface places in the subnet which is being reduced, therefore we maintain the marking bound of places. Thus, the liveness of the system is also preserved. ∎

9.3 Quantitative Aspects

During the FEA aggregation, we replace a subnet SN_2 by a flow equivalent net (Definition 9.2.2). In the quantitative analysis, we have to determine the conditional throughput through the output interface places of SN_2 as a function of customers n in the system, $n = 1 \ldots K$. The subnet is analyzed in short circuit by feeding the tokens which exit through the output interface back into the input interface via the immediate transitions $t_{21}^1 \ldots t_{21}^r$ as shown in Figure 9.4. This way we keep the number of customers in SN_2 constant.

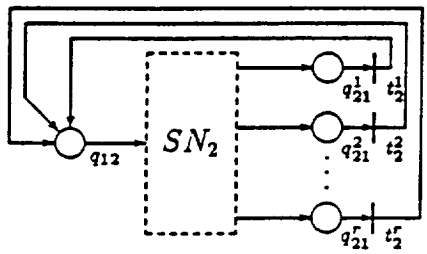


Figure 9.4: Feedback Construct for Flow Equivalent Net $\mathcal{N}_{\text{FEN}}$

Rule 4: SIMO-MPMT Feedback Construct Let $\tilde{t_2} = \{t_2^1 \ldots t_2^r\}$ as shown in Figure 9.4. The feedback construct is given by:

1. Structure: $\mathcal{N}_{\text{FEN}} = (\tilde{P}_2,\ \tilde{T},\ \tilde{F})$, is a feedback construct, where:

   (a) $\tilde{P} = P_2$

   (b) $\tilde{T} = T_2 \cup \tilde{t_2}$

(c) $\tilde{F} = F_2 \cup (Q_{21} \times \tilde{t_2}) \cup \tilde{t_2} \times \{q_{12}\})$; $(\Rightarrow q_{12} = \tilde{t_2}^{\bullet}, Q_{21} =^{\bullet} \tilde{t_2})$

i.e., ${}^{\bullet}t_2^i = q_{21}^i$, $t_2^{i\bullet} = q_{12}$, $i = 1 \ldots r$ as shown in Figure 9.4

2. Marking:

   Let $\overline{M}$ be a solution to the linear programming problem LPP6. The initial marking for the SIMO-MPMT Feedback Construct is the projection of $\overline{M}$ on $\mathcal{N}_{\text{FEN}}$ with the exception of q_{12} (the marking of a single interface will be used to vary the number of customers in the system).

We are now ready to formulate the algorithm for the quantitative analysis:

Algorithm 3: Quantitative Analysis of Well-Formed SIMO-MPMT Cuts

1. Select a well formed SIMO-MPMT cut ξ
2. Generate the flow aggregated net systems $\mathcal{FS}$ (Figure 9.3) by using the SIMO-FEA Reduction Rule and the feedback construct by using the SIMO-MPMT Feedback Construct Rule as shown in Figure 9.4.
3. Analyze the feedback construct in Figure 9.4 for $n = 1 \ldots K$ and obtain the conditional throughputs $\mathcal{X}^i(n)$ of t_2^i, $i = 1 \ldots r$ and $n = 1 \ldots K$.
4. In $\mathcal{FS}$, set the the firing rates $\mu_2^i(n)$ equal to the conditional throughputs obtained in step 3, i.e.,

$$\mu_2^i(n) = \mathcal{X}^i(n), \quad i = 1 \ldots r, \quad n = 1 \ldots K \tag{9.2}$$

5. Analyze $\mathcal{FS}$ by solving the underlying CTMC and obtain the desired performance measures.

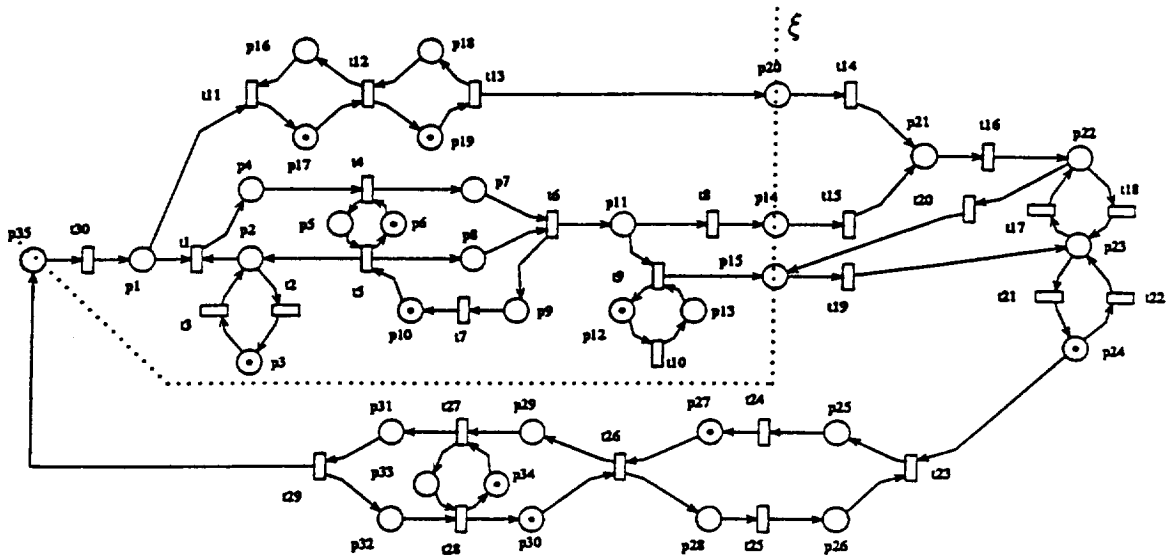


Figure 9.5: Example of MPMT-Net with Indicated Well-Formed SIMO-Cut. $q_{12} = P35$, $Q_{21} = \{p_{20}, p_{14}, p_{15}\}$

9.4 Example for SIMO-MPMT Cut

Figure 9.5 shows the example of an MPMT-net (but not FC) without a physical interpretation. The dashed box indicates the SIMO cut and the resulting SIMO subnet SN_2. Let us first verify that ξ is indeed well-formed by eliminating the macroplaces and macrotransitions until SN_2 is reduced to a state machine. By visually examining the net (or by using the algorithm outlined in [Sil81]), it is straight forward to check, that $MT_1 = \{t_{11}, t_{12}, t_{13}\}$ and $MT_2 = \{t_9, t_{10}\}$ form macrotransitions. After the reduction of the macroplace $\{p_2, t_2, p_3, t_3\}$, the set of transitions formed by $MT_3 = \{t_1, t_4, t_5, t_6, t_7\}$ also forms a macrotransition. Figure 9.6 shows the *structure* of the resulting state machine. Note that the indicated cut is the only possible well-formed cut to the system.

Let us now step through Algorithm 3: *Quantitative Analysis of Well-Formed SIMO-MPMT Cuts*:

1. The cut is already selected and well-formed.

2. Before generating the flow aggregated system $\mathcal{FS}$ and the Feedback Construct,

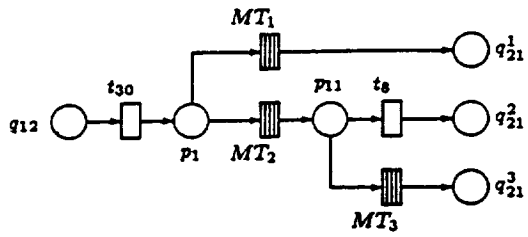


Figure 9.6: Transformed Subnet SN_2 in State Machine Form (Macrotransitions are Shown Dashed)

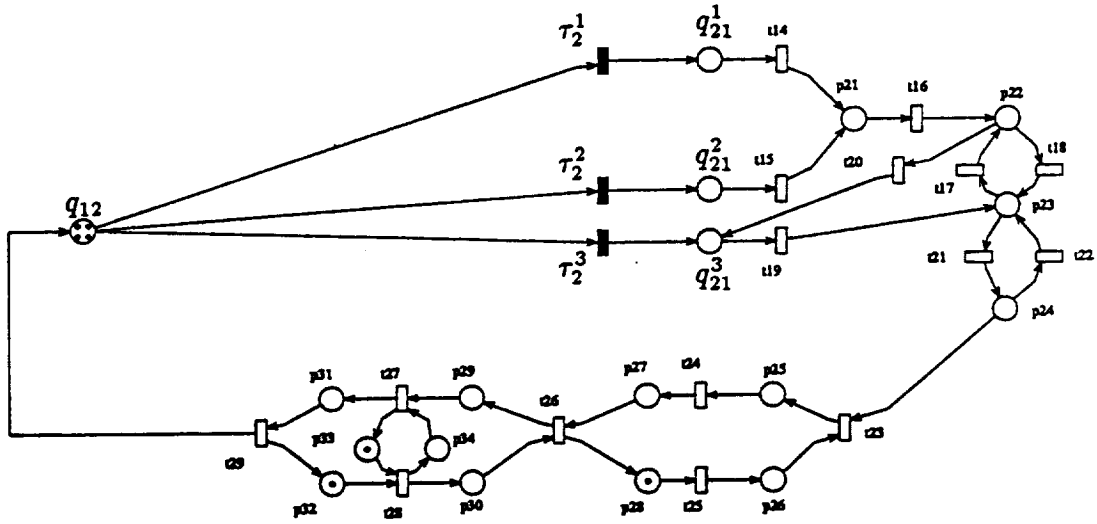


Figure 9.7: Flow Aggregated System $\mathcal{FS}$ for MPMT-Net with SIMO-Cut ($K = \overline{M}(q_{12}) = 4$)

we have to solve LPP6. A possible solution is:

$$M(p_{35}) = 4,\ M(p_3) = M(p_6) = M(p_{10}) = M(p_{13}) \tag{9.3}$$

$$= M(p_{16}) = M(p_{18}) = M(p_{28}) = M(p_{32}) = M(p_{33}) \tag{9.4}$$

$\mathcal{FS}$ and the Feedback Construct with the computed initial marking from LPP6 are shown in Figure 9.7 and 9.8 respectively. Now the number of customers $M(q_{12}) = M(p_{35}) = 4$.

3. We now vary the number of customers in the system from $n = 1 \ldots 4$. Table 9.1 shows the conditional throughputs of $\tilde{t}_2$ obtained by varying the number of

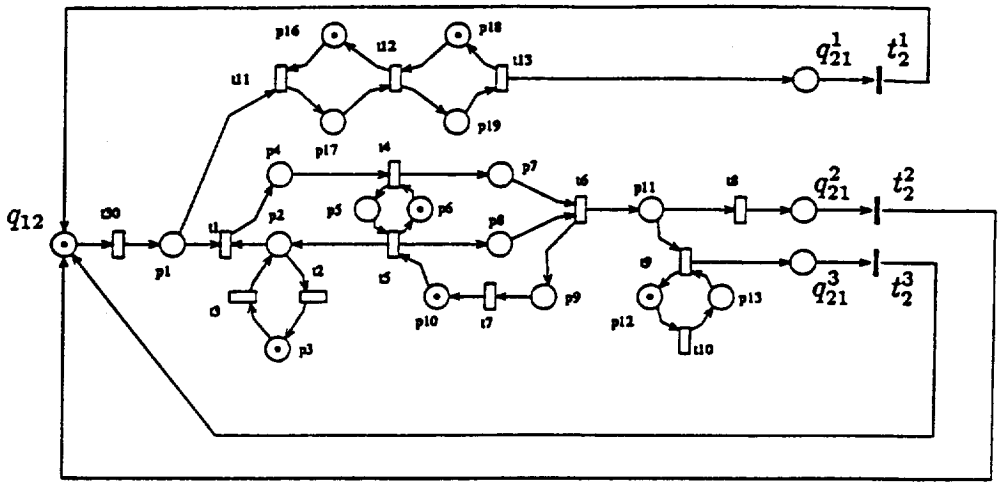


Figure 9.8: Feedback Construct for MPMT-Net with SIMO-Cut (Number of Customers Shown: $M(q_{12}) = 1$)

customers in the system, i.e., by varying $M_{0q12} = 1 \ldots K$, as well as the tangible state space (column 5).

	$\mathcal{X}_2^1$	$\mathcal{X}_2^2$	$\mathcal{X}_2^3$	tang. state space
1	0.167907	0.036307	0.036105	52
2	0.274374	0.066655	0.065204	168
3	0.337118	0.088924	0.085549	376
4	0.370327	0.102093	0.097113	680

Table 9.1: Conditional Throughputs using Flow Equivalent Aggregation for SIMO Cut at MPMT-net

4. In this step we substitute the conditional throughputs obtained in step 3 back into $\mathcal{FS}$.

5. Table 9.2 shows the results of the aggregation.

By aggregating SN_2 to a FEN, we have introduced very little error, while at the same time reducing the tangible state space by a factor of roughly 37.

	χ_2^1	χ_2^2	χ_2^3	tang. state space
FEA	0.109755	0.024093	0.024274	88720
exact	0.108199	0.024497	0.024191	2424
error	1.4 %	-1.64 %	0.3 %	—

Table 9.2: Final Results Using Flow Equivalent Aggregation for SIMO Cut at MPMT-net

9.5 Summary

In this chapter, we have introduced a new class of cuts to MPMT-nets, where we allow the particular case of a multiple output interface. We require that the interface has the characteristics of a server in a network that allows Product Form Decomposition which was introduced in Definition 2.3.1.

We have generalized the SISO interface for FEA to flow equivalent nets which capture the input/output behavior of the SIMO subnet. For the shown example, the accuracy of the method is good with a large reduction in the state space. However, it has to be kept in mind, that good accuracy can by no means be guaranteed especially if there exists strong dependence of the MCT on the interarrival times or large coefficient of variation in the service time of transitions.

CHAPTER 10

Case Study of a Dual Arm Robotics System for Assembly

10.1 Introduction

In this chapter we present an application from the area of robotics as applied to space activities. The demo consists of the insertion of a strut into a structure (i.e., two previously assembled struts). We will first describe the physical and hardware configuration of the system. Then we will describe (in words) the task at hand and present the Petri net model. We then apply flow equivalent aggregation which was presented in the previous chapters.

The following description of the hardware and software components at the Center for Intelligent Robotics for Space Exploration (CIRSSE) is adapted from [LS91].

The *Theory of Intelligent Machines*, as proposed in [Sar79], describes a hierarchical organization of the functions of an autonomous robot into three levels: on the lowest level is the *Execution Level* containing the hardware and basic control functions, further up in the hierarchy is the *Coordination Level* integrating the capabilities of the intelligent machine across hardware systems, and an *Organization Level* providing higher-level planning and reasoning capabilities. The nature of the intelligent machine levels are different: The Execution Level provides the control functions, the Coordination Level is characterized by scheduling and operations research issues, and the Organization Level employs artificial intelligence techniques for task planning. A diagram of the intelligent machine is shown in Figure 10.1. The model is organized according to the principle of *Increasing Precision With Decreasing Intelligence* [SV77, SV88]. Detailed run-time information is maintained at the lowest Execution Level and only relevant status information is propagated upward

through the hierarchy.

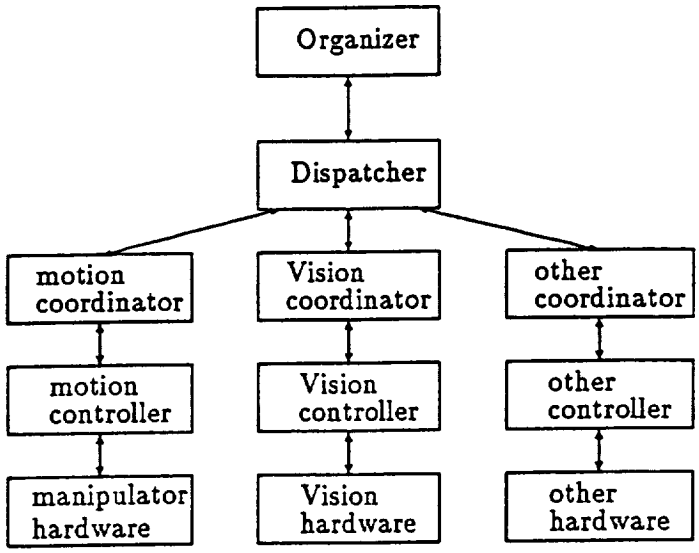


Figure 10.1: Logical Model of Intelligent Machine

In the case study, the entire strut assembly operation is started by a single **assemble_strut** command from the Organization Level to the Coordination Level Dispatcher. Therefore, operation of the Organization Level is omitted from this case study and only the execution and coordination level are considered.

10.2 Hardware and Software Configuration

The testbed hardware at CIRSSE consists of an 18-DOF manipulator system, a 5 camera vision subsystem, their associated control computers, and several UNIX workstations. The 18-DOF manipulator is built from two 6-DOF PUMA robots (a PUMA 560 and a PUMA 600) mounted on a custom-built robot transporter with two 3-DOF carts. The carts travel along a 12 foot linear track and provide $\pm 45°$ of tilt and $\pm 150°$ of rotation. This manipulator subsystem is controlled by a *Motion Control System*. For the experiment at hand, only one of the manipulators was used.

The vision subsystem has five cameras that may be used individually or pairwise, a laser scanner for pointing light, and a (VME-based) Datacube image processing system. Additional processing capabilities are provided by four UNIX workstations (predominantly Sun 4 and Sparc Stations) attached to an Ethernet which connects all of the testbed subsystems.

As shown in Figure 10.1, the Coordination Level of the intelligent machine is organized into a tree structure, with a *Dispatcher* at the root and multiple *Coordinators* at the leaves.

The Dispatcher is the first component of the intelligent machine that deals with the machine as a whole. It coordinates actions across Execution Level functions, e.g., move a manipulator to a position determined by visual sensing. The Dispatcher defines the executions of the subtasks.

Petri nets are used to implement the event-driven environment of the dispatcher and the coordinators [WS90]. In this case study, we concentrate on the level of the hierarchy represented by the dispatcher.

10.3 Case Study

As a case study, the dispatcher realized the insertion of a strut into an existing V-node. A schematic diagram of the experimental setup is shown in Figure 10.2. In addition to the two cameras mounted on the wrist of the robot, two additional cameras at the ceiling are used, which is not shown in the figure.

The task is as follows: After the initialization of the robot and the vision system, the cameras mounted on the ceiling find the preassembled V-struts (essentially two struts from a previous assembly operation). The robot then moves to the end nodes of the V-strut lying on the table and determines the exact location of the insertion point with the wrist cameras. The robot then proceeds to move to the rack and picks up the strut to be inserted. After the pickup operation, the robot

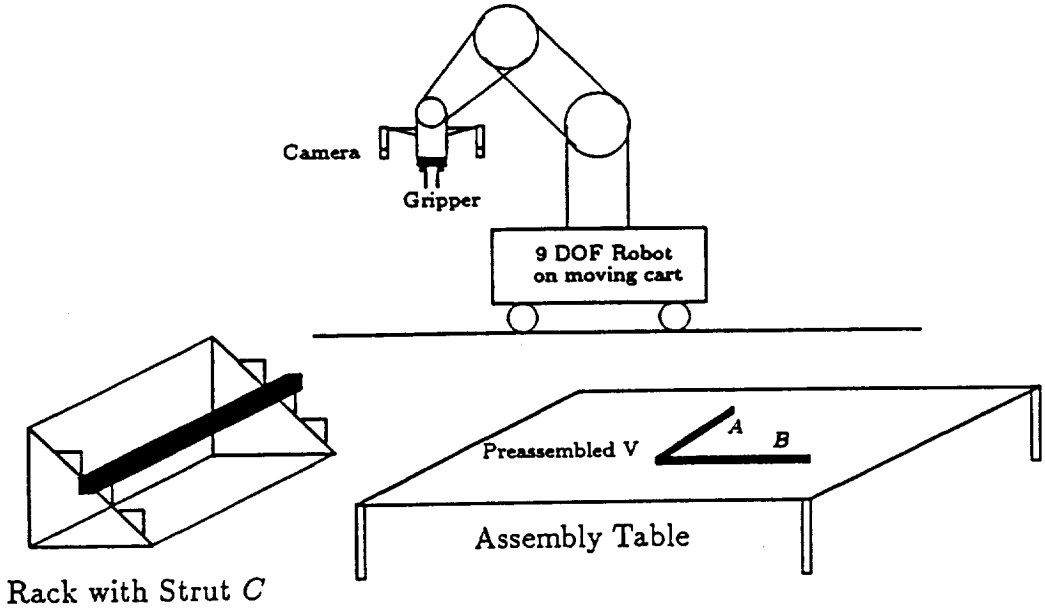


Figure 10.2: Setup for Strut Insertion

moves back to the table and inserts the strut into the V-strut to form a triangle.

Figure 10.3 shows a high level representation of the dispatcher. The four Macrotransitions represent the following activities:

MT_1: Initialize the robot and vision system and localize the V-strut (i.e., strut A and B which have previously been assembled) with the ceiling camera.

MT_2: Move to the nodes of the V-strut and use the wrist camera to obtain the exact position.

MT_3: Move to the rack, localize both ends of strut C with the wrist camera in order to obtain its exact position.

MT_4: Pick up strut C and insert it into the V-strut. Shut down the system.

As can be easily checked, the net in Figure 10.3 is not strongly connected. We obtain a strongly connected net (an MPMT-net) by merging the places *req_demo* and *demo_complete*. The net is now repetitive and the performance analysis can be

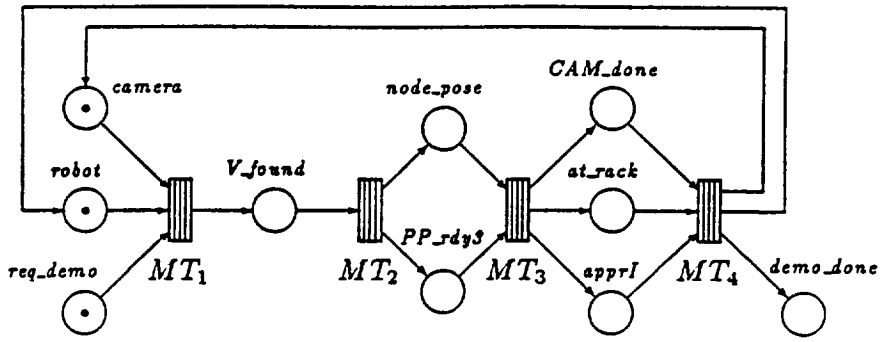


Figure 10.3: High Level Representation of Dispatcher

done. The obtained cycle time is the required time from the start to the finish of the activities. Figure 10.4 shows a detailed representation of the aggregated transitions. Note that all shown subnets with the exception of MT_1 are macrotransitions. In MT_1, place *PP_ready* is a choice place. Transitions *homed* and *not_home* have *enabling functions* attached to them, i.e., they will fire depending on the position of the robot. Here we assume, that the robot is already homed before the demonstration. Therefore, the resulting model is a MG. The complete net can be obtained by merging MT_1 through MT_4 at the places which bear the same label. As shown in the figure, only places *camera* and *req_demo* are marked.

Table 10.1 gives a listing of all transitions, together with a brief description of the associated activity and the required time to completion (i.e., the mean interfiring time). The values were obtained experimentally, by stepping the testbed through the assembly operations and measuring the times.

The marking of the dispatcher is safe (i.e., 1-bounded) and no internal circuits are present. We therefore choose flow equivalent aggregation for the approximate analysis of the system. The dispatcher is partitioned into four subsystems MT_i, $i = 1 \ldots 4$. All cuts to the system are SISO. After the canonical transformation, a solution to the linear programming problem LPP3 shows, that each of the subnets

should be analyzed with a single token.

Table 10.2 shows the results of the analysis. The original net has 62 tangible states. The mean time to complete the demo was measured to be 207 seconds, while the mean time as obtained by solving the underlying CTMC of the Petri net system is 205.7 seconds. This is also the results of the flow equivalent aggregation therefore, in this particular case the aggregation is exact.

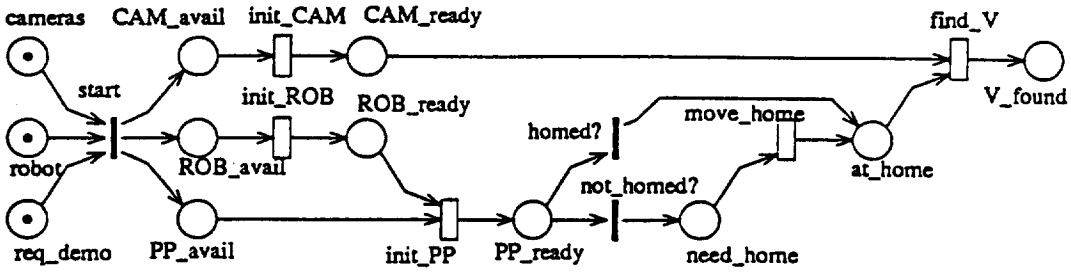
10.4 Summary

From this case study, we can draw two conclusions:

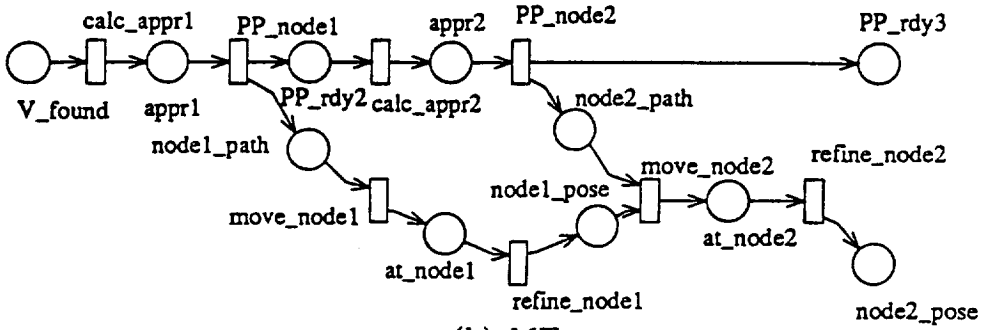
- The agreement of the measured value of the completion time of the demo with the value obtained by solving the CTMC is good. This substantiate the fact, that we can use the stochastic Petri net model for the performance analysis.

- In certain cases, FEA can lead to *exact results.* This comes as no surprise. In the example in Section 5.2.3 we have already encountered a case where the aggregation was exact. In that case, no internal loops or trapped tokens were present, furthermore the marking was safe. This is exactly the same case here. In this particular case, the MCT is independent of the interarrival process. Furthermore, in cyclic queues with exponential servers and a single customer, the cycle time is just the sum of the service times of the servers.

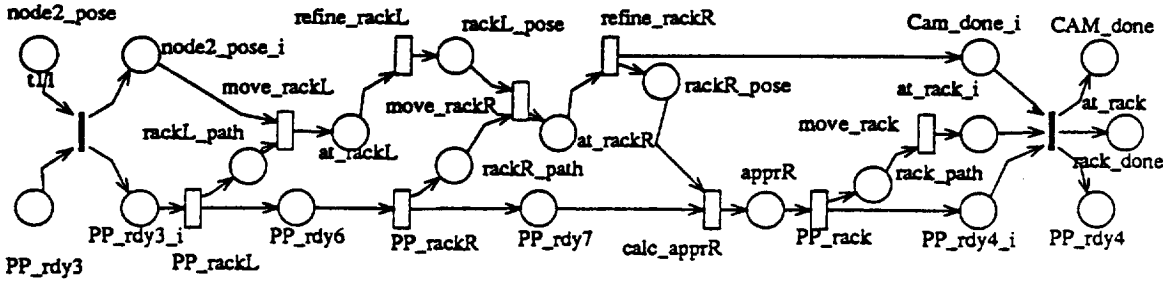
The analyzed case study is the simple task of inserting a strut into a previously existing structure. In the future, the testbed system will perform far more complicated tasks. Therefore, the dispatcher will be more complex and its associated state space will grow large. The performance analysis will become increasingly difficult, because of the large state space. By using the presented methodology, we can predict the performance (i.e., the time required to execute a task) before the system is actually implemented.



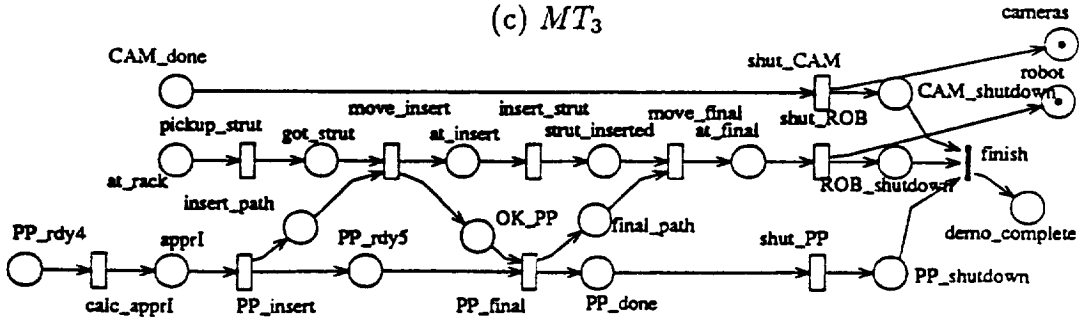
(a) MT_1



(b) MT_2



(c) MT_3



(d) MT_4

Figure 10.4: Exploded Composite Transitions MT_1, MT_2, MT_3 and MT_4 of CIRSSE Testbed

Interfiring times in seconds		
Transition	time	Description
start	0.0	start the demo (immediate)
init_CAM	0.8	initialize cameras (wrist and ceiling)
init_ROB	11.2	initialize robot
init_PP	1.0	initialize path planner
move_home	10.0	move robot to home position
find_V	10.4	find V-strut with ceiling camera
calc_appr1	0.8	compute approach to V-strut (node 1)
PP_node1	2.0	plan path to node 1
move_node1	13.3	move robot to node 1
calc_appr2	1.3	compute approach to V-strut (node 2)
refine_node1	5.8	locate node 1 with wrist camera and refine location
PP_node2	1.6	plan path to node 2
move_node2	10.3	move robot to node 2
refine_node2	5.8	locate node 2 with wrist camera and refine location
rack_rdy	0.0	Ready to start activities at rack (immediate)
PP_rackL	1.7	plan path to left node of strut on the rack
move_rackL	20.0	move robot to left node of strut on the rack
refine_rackL	6.6	locate left node of strut on the rack with wrist camera
PP_rackR	1.5	plan path to right node of strut on the rack
move_rackR	10.0	move robot to right node of strut on the rack
refine_rackR	7.2	locate right node of strut on the rack with wrist camera
calc_apprR	1.7	calculate approach for pickup of strut
PP_rack	1.7	plan path for pickup of strut
move_rack	7.2	move robot to rack
rack_done	0.0	activities at rack done (immediate)
pickup_strut	9.0	pickup strut from rack
calc_apprI	1.9	calculate approach for insert of strut into V-strut
PP_insert	1.7	plan path for insert of strut
move_insert	20.8	move robot to insertion point
insert_strut	34.0	insert strut into V-strut
PP_final	2.0	plan path to final position of robot
move_final	10.7	move robot to final position
shut_CAM	0.5	shut down vision system
shut_ROB	13.4	shut down robot
finish	0.0	everything finished

Table 10.1: Description and Interfiring Times of Transitions in Dispatcher (Experimental Data)

Time unit in seconds			
aggr. net	$\mathcal{X}$	Γ	tang. state space
MT_1	0.044192	22.6283	8
MT_2	0.026242	38.1068	12
MT_3	0.017819	56.1199	11
MT_4	0.011253	88.8652	31

Table 10.2: Results of the Flow Equivalent Aggregation of the Subnets Composing the CIRSSE Testbed

CHAPTER 11
Conclusions

11.1 Discussion

In the present work, we have introduced the structural decomposition of Marked Graphs (MGs) and Macroplace/Macrotransition nets (MPMT-nets), a subclass of Petri nets which allows limited choice and synchronizations. Single cuts lead to aggregated net systems, where one of the subnets is reduced to a single transition and the basic skeleton, where both subnets are reduced to a transition.

The cuts can be categorized according to their interface places, i.e., places which both subnets have in common. Single Input/Single Output (SISO) cuts do not introduce any synchronization constraints or marking bound changes and are therefore preferred.

In MGs, Single Input/Multiple Output (SIMO/MISO) and Multiple Input/Multiple Output (MIMO) cuts introduce additional synchronization constraints, and can lead to a different net behavior. In some cases, the synchronization of concurrent processes can lead to bad results.

The introduced cuts (with the exception of the SIMO cut in MPMT-nets) lead to a product form representation of the basic skeleton, i.e., it can be solved by analytic means (e.g., Mean Value Analysis).

Once the net is decomposed, a variety of methods can be applied for the approximate analysis.

- **Response Time Approximation:** [DJS92, JSS92b]

  This method is introduced in the present work. From the analysis of the aggregated net system, RTA only uses the throughput of the analysis of the aggregated net system. Other information such as the token distribution in

the subnet is not used. RTA is very robust, converges quickly and shows good accuracy.

- **Marie's Method:** [Mar79]

  In Marie's method, we use information about the token distribution in the aggregated net systems. In all considered SISO cuts, the additional information leads to an improved accuracy over RTA, although convergence is usually slower. In one particular case, Marie's method failed to converge.

- **Flow Equivalent Aggregation (FEA):** [JD91a, LZGS84]

  FEA is computationally very efficient, but has a severe drawback. If the mean completion time of a subnet depends strongly on the interarrival process, the aggregation can lead to very bad results as seen in Section 8.2.2. FEA should be used with caution, and the subnets should be *tested* whether a strong dependence of the mean completion time on the arrival process exists (as outlined in Section 5.2.3).

- **Delay Equivalence (DE):** [LW91, WL91]

  Delay equivalence was introduced in the context of Petri nets by Woodside et al. State dependent DE shows very poor convergence. In the presented examples, this method failed in most cases. State independent DE converges more often, but still did not reach a fixed point solution in all cases. Furthermore, the number of iterations is high compared to RTA or Marie's method. When DE converges, the accuracy is similar to RTA. In DE, information about the throughput and the token distribution in both subnets is used.

Based on our experimental results, we can now give guidelines for using the different methods:

- *Delay Equivalence* shows no advantage over any of the other methods and should not be used.

- *Flow Equivalent Aggregation* should only be used when no internal cycles are present, or it has been verified that the mean completion time of the subnet does not depend on the arrival process (Section 5.2.3). In any case, FEA can yield good initial values for Marie's method and RTA.

- *Marie's Method* is the method of choice when single SISO cuts are considered. Although convergence is often slower than in RTA, the accuracy was better in all considered single SISO cuts.

- *Response Time Approximation* should be used when FEA is not applicable, computation time is at a premium or Marie's does not converge. This represents the classical cost versus quality tradeoff.

Unfortunately, for the hierarchical decomposition we are not able to give such guidelines as no scheme is consistently better.

In MPMT-nets, FEA was extended to SIMO cuts. This allowed the formulation of a flow equivalent server which captures the input/output behavior of the subnet.

As a real life case study, a dual arm robotics testbed for assembly operations was analyzed. The results show, that the proposed aggregation is suitable in that context.

11.2 Outlook

Based on the present work, we can identify several research directions:

1. If the customer load at the cut is low (e.g., $K = 1$), the presented methods can lead to bad results. One approach could be to capture the higher moments of the probability distribution of the residence time of the tokens in the interface places. This way, we would be able to use more information about the aggregated net systems, which could lead to a better approximation.

2. At present, the choice of the cuts is limited. By devising different basic skeletons, which take the multiple nature of the interface into account, we could allow more general types of cuts.

3. The almost product form decomposition in Petri nets can be further explored. A first step in that direction was done in Chapter 9, where SIMO cuts in MPMT-nets were introduced.

LITERATURE CITED

[ABS84] S. C. Agrawal, J. P. Buzen, and A. W. Shum. Response time preservation: a general technique for developing approximate algorithms for queueing networks. In *Proceedings of the 1984 ACM Sigmetrics Conference on Measurment and Modeling of Computer Systems*, 1984.

[AJ88] R. Y. Al-Jaar. *Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri Nets.* PhD thesis, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, 1988.

[AJD90a] R. Y. Al-Jaar and A. A. Desrochers. *Advances in Robotics and Automation*, volume 2, Saridis, G. N. (ed), chapter Petri Nets in Automation and Manufacturing, pages 153–225. JAI Press, 1990.

[AJD90b] R. Y. Al-Jaar and A. A. Desrochers. Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets. *IEEE Transactions on Robotics and Automation*, 6(6):621–639, December 1990.

[AMBC+89] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, 7(15):832–846, July 1989.

[BBG86] G. Balbo, S. C. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena. *IEEE Transactions on Software Engineering*, 12(4):561–576, April 1986.

[BBG88] G. Balbo, S. C. Bruell, and S. Ghanta. Combining queueing networks and generalized stochastic Petri net models for the solution of complex models of system behaviour. *IEEE Transactions on Computers, Vol. 37, No. 10*, 37(10):1251–1268, 1988.

[BC90] G. Balbo and G. Chiola. Stochastic Petri net simulation. In *Proceedings of the 1990 Winter Simulation Conference*, pages 266–276, 1990.

[BCFR87] G. Balbo, G. Chiola, G. Franceschinis, and G. M. Roet. Generalized stochastic Petri nets for the performance evaluation of FMS. In

Proceedings of the 1987 IEEE International Conference on Robotics and Automation, pages 191–198, Raleigh, North Carolina, 1987.

[BCMP75] F. Baskett, M. K. Chandy, R. R. Muntz, and Fernando G. Palacios. Open, closed and mixed networks of queues with different class of customers. *Journal of the Association of Computing Machinery*, 22(2):248–260, 1975.

[BCR87] G. Balbo, G. Chiola, and M. G. Roet. On the efficient construction of the tangible reachability graph of generalized stochastic Petri nets. In *Proceedings of the Second International Workshop on Petri Nets and Performance Models*, pages 136–145, Madison, Wi, 1987. IEEE Computer Society Press.

[BD90] B. Baynat and Y. Dallery. A unified view of product-form approximation techniques for general closed queueing networks. Technical Report 90.48, Lab. MASI, Université P. et M. Curie (Paris 6), Paris, France, October 1990.

[Beo85] C. Beounes. Stochastic Petri net modeling for dependability evaluation of complex computer systems. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 191–198, Torino, Italy, July 1985.

[Ber85] G. Berthelot. Checking properties of nets using transformations. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets'85*, volume 222 of *LNCS*, pages 19–40. Springer-Verlag, 1985.

[Bra85] A. Brandwajn. Equivalence and decomposition in queueing systems - a unified approach. *Performance Evaluation*, 5:175–186, 1985.

[BT86] A. Bobbio and K. S. Trivedi. An aggregation technique for the transient analysis of stiff Markov chains. *IEEE Transactions on Computers*, C-35(9):803–814, 1986.

[Cam90] J. Campos. *Performance Bounds for Synchronized Queueing Networks.* PhD thesis, Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, Spain, December 1990. Research Report GISI-RR-90-20.

[Cam91] J. Campos. Performance analysis of live and bounded free choice systems. In E. Best and J. Esparza, editors, *Design Methods Based on Nets*, pages 38–48. GMD-Studien Nr. 198, Sankt Augustin, Germany, September 1991.

[CCCS89] J. Campos, G. Chiola, J. M. Colom, and M. Silva. Tight polynomial bounds for steady-state performance of marked graphs. In

Proceedings of the Third International Workshop on Petri Nets and Performance Models, pages 200–209, Kyoto, Japan, December 1989. IEEE Computer Society Press.

[CCCS92] J. Campos, G. Chiola, J. M. Colom, and M. Silva. Properties and performance bounds for timed marked graphs. *IEEE Transactions on Circuits and Systems*, 39(5), May 1992.

[CCS90] J. Campos, J. M. Colom, and M. Silva. Performance evaluation of repetitive automated manufacturing systems. In *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pages 74–81, Rensselaer Polytechnic Institute, Troy, NY, USA, May 1990. IEEE Computer Society Press.

[CCS91a] J. Campos, G. Chiola, and M. Silva. Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector. *IEEE Transactions on Software Engineering*, 17(2):117–125, February 1991.

[CCS91b] J. Campos, G. Chiola, and M. Silva. Properties and performance bounds for closed free choice synchronized monoclass queueing networks. *IEEE Transactions on Automatic Control*, 36(12):1368–1382, December 1991.

[CG87] Y. F. Choong and S. B. Gershwin. A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times. *IIE Transactions*, 19(2):150–159, June 1987.

[Chi85] G. Chiola. A software package for the analysis of generalized stochastic Petri nets. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 136–143, Torino, Italy, July 1985.

[Cou75] P.J. Courtois. Decomposability, instabilities and saturation in multiprogramming systems. *Communications of the ACM*, 18(5):371–377, July 1975.

[Cou77] P.J. Courtois. *Decomposability, Queuing and Computer System Applications.* Academic Press, 1977.

[CS78] M. K. Chandy and C. H. Sauer. Aproximate methods for analyzing queueing network models of computing systems. *Computing Surveys*, 10(3):281–317, September 1978.

[CS90] J. M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets'90*, volume 483 of *LNCS*, pages 113–145. Springer-Verlag, 1990.

[CT91a] 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, Australia, 1991. IEEE Computer Society Press.

[CT91b] G. Ciardo and K. S. Trivedi. Solution of large GSPN models. In W. Stewart, editor, *Numerical Solution of Markov Chains*. Marcel Dekker, New York, 1991.

[DBCT85] J. B. Dugan, A. Bobbio, G. Ciardo, and K. Trivedi. The design of a unified package for the solution of stochastic Petri net models. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 6–13, Torino, Italy, July 1985.

[DDL89] Y. Dallery, R. David, and Xie X. L. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 34(9):943–953, September 1989.

[Deo74] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[DG91] Y. Dallery and S. B. Gershwin. Manufacturing flow line systems: A review of models and analytical results. Technical Report 91.18, Lab. MASI, Université P. et M. Curie (Paris 6), Paris, France, March 1991.

[DJS92] A.A. Desrochers, H. Jungnitz, and M. Silva. An approximation method for the performance analysis of manufacturing systems based on GSPNs. In *Proceedings of Renssalear's Third International Conference on Computer Integrated Manufacturing*, 1992.

[DLT90] Y. Dallery, Z. Liu, and D. Towsley. Equivalence, reversibility and symmetry properties in fork/join queueing networks with blocking. Technical Report M.A.S.I. 90.32, Lab. MASI, Université P. et M. Curie (Paris 6), Paris, France, = June 1990.

[DLT91] Y. Dallery, Z. Liu, and D. Towsley. Reversibility in fork/join queuing networks with blocking after service, 1991. In preparation.

[ES90a] J. Esparza and M. Silva. On the analysis and synthesis of free choice systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1987*, pages 243–286. Springer-Verlag, 1990.

[ES90b] J. Esparza and M. Silva. Top-down synthesis of live and bounded free choice nets. In *Proceedings of the 11th International Conference on Application and Theory of Petri Nets*, pages 63–83, Paris, France, June 1990.

[GB81] S.B. Gershwin and O. Berman. Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers. *AIIE Transactions*, 13(1):2–11, March 1981.

[GDZ90] D. Guo, F. DiCesare, and M. Zhou. A monent generating function based approach for evaluating arbitrary stochastic Petri nets, 1990. Unpublished.

[GH85] D. Gross and C.M. Harris. *Fundamentals of Queueing Theory.* John Wiley & Sons Inc., 1985.

[GN67] W. J. Gordon and G. F. Newell. Closed queuing systems with exponential servers. *Operations Research*, 15:254–265, 1967.

[Hac72] M.H.T. Hack. Analysis of production schemata by Petri nets. Technical Report TR-94, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1972.

[HC79] Y. C. Ho and T. T. Chien. A gradient technique for general buffer storage design in a production line. *International Journal of Production Research*, 17(6):557–580, 1979.

[HC83] Y. C. Ho and T. T. Chien. A new approach to determine parameter sensitivities of transfer lines. *Management Science*, 29(6):700–714, 1983.

[HPTD90] W. Henderson, C. M. Pearce, P. G. Taylor, and N. M. Djik. Closed queueing networks with batch services. *Queueing Systems*, 6:59–70, 1990.

[HS89] P.J. Haas and G.S. Shedler. Stochastic Petri net representation of discrete event simulation. *IEEE Transactions on Software Engineering*, 15(4):381–393, April 1989.

[HT90] W. Henderson, , and P. G. Taylor. Product form networks of queues with batch arrivals. *Queueing Systems*, 6:71–87, 1990.

[HV85] M. A. Holliday and M. K. Vernon. A generalized Petri net model for performance analysis. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 181–190, Torino, Italy, July 1985.

[Jac57] J. R. Jackson. Network of waiting lines. *Operations Research*, 5:518–521, 1957.

[Jac63] J. R. Jackson. Jobshop like queueing systems. *Management Science*, 10:131–142, 1963.

[JD91a] H. Jungnitz and A. A. Desrochers. Flow equivalent nets for the performance analysis of flexible manufacturing systems. In *Proceedings of the IEEE Robotics and Automation Conference 1991*, pages 122–127, 1991.

[JD91b] H. Jungnitz and A. A. Desrochers. State space reduction techniques for discrete event dynamic systems. In *Proceedings of the IMACS Symposium MCTS 1991*, pages 799–805, Lille, France, 1991.

[JGD91] H. Jungnitz, A. Giua, and A. Desrochers. An integration based method for the solution of continuous time Markov chains. In *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 385–390, Charlottesville, VA, USA, 1991.

[JSS92a] H. Jungnitz, B. Sanchez, and M. Silva. Response time approximation for the performance analysis of manufacturing systems modeled with stochastic marked graphs. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[JSS92b] H. Jungnitz, B. Sanchez, and M. Silva. Response time approximation for throughput computation of stochastic marked graphs. *Journal of Parallel and Distributed Computing*, 1992. Accepted.

[Kar84] N. Karmakar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–507, 1984.

[KBB87] K.M. Kavi, B.P. Buckles, and U.N. Bhat. Isomorphism between Petri nets and dataflow graphs. *IEEE Transactions on Software Engineering*, 13(10):1127–1133, October 1987.

[KDD88] E. Kasturia, F. DiCesare, and A. Desrochers. Real time control of multilevel manufacturing systems using coloured Petri nets. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1114–1119, Philadelphia, PA, 1988.

[Kle75] L. Kleinrock. *Queueing Systems*, volume 1,2. Wiley Interscience Publications, 1975.

[Koe59] E. Koenigsberg. Production lines and internal storage - a review. *Management Science*, 5:410–433, 1959.

[LS91] D.R. Lefebvre and G.N. Saridis. A computer architecture for intelligent machines. Technical Report CIRSSE-108, Rensselaer Polytechnic Institute, Center for Intelligent Robotics for Space

Exploration (CIRSSE), Rensselaer Polytechnic Institute, Troy, NY 12180-3590, December 1991.

[LW91] Y. Li and C. M. Woodside. Iterative decomposition and aggregation of stochastic marked graph Petri nets. In *Proceedings of the 12th International Workshop on Application and Theory of Petri Nets*, Aarhus, Denmark, June 1991.

[LZGS84] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative Systems Performance: Computer System Analysis Using Queueing Network Models.* Prentice Hall, Inc., 1984.

[Mar71] C. W. Marhall. *Applied Graph Theory.* John Wiley & Sons, New York, 1971.

[Mar79] R. Marie. An approximate analytical method for general queueing networks. *IEEE Transactions on Software Engineering*, SE-5(5):530–538, September 1979.

[Mar89] A. Marsan. Stochastic Petri nets: An elementary introduction. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1989*, pages 1–29. Springer-Verlag, 1989.

[MFDD89] M. Di Mascolo, M.Y. Frein, Y. Dallery, and R. David. Modeling of Kanban systems using Petri nets. In *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, pages 307–312. Elsevier Science publishers B.V. (North Holland), 1989.

[MJ88] M.Silva and J.M.Colom. On the computation of structural synchronic invariants in p/t nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets'88 - Part I*, volume 340 of *LNCS*, pages 386–417. Springer-Verlag, 1988.

[Mol82] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers, Vol. C-31, No. 9*, pages 913–917, 1982.

[Mur77] T. Murata. Circuit theoretic analysis and synthesis of marked graphs. *IEEE Transactions on Circuits and Systems*, 24(7):400–405, July 1977.

[Mur89] T. Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[NRKT89] G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd. *Handbooks in Operation Research and Management Science.* North-Holland, 1989.

[OP86] R. Onvural and H. G. Perros. On equivalencies of blocking mechanism in queuing networks with blocking. *Operation Research Letters*, 5-6:293–298, 1986.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Ram74] C. Ramchandani. *Analysis of asynchronous Concurrent Systems by Petri nets.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1974.

[RL80] M. Reiser and S.S. Lavenberg. Mean value analysis of closed queueing networks. *Journal of the Association for Computing Machinery*, 27(2):313–323, 1980.

[Ros83] S.M. Ross. *Stochastic Processes.* John Wiley & Sons, New York, 1983.

[RP84] R. R. Razouk and C. V. Phelps. Performance analysis using timed Petri nets. In *Proceedings of the International Conference on Parallel Processing*, pages 126–128. IEEE Computer Society Press, 1984.

[Sar79] G.N. Saridis. Toward the realization of intelligent controls. *Proceedings of the IEEE*, 67(8):34–39, 1979.

[Sil81] M. Silva. Sur le concept de macroplace et son utilisation pour l'analyse des reseaux de petri. *RAIRO-Systems Analysis and Control*, 15(4):57–67, 1981.

[Sil85] M. Silva. *Las Redes de Petri: en la Automatica y la Informatica.* Editorial AC, Madrid, Spain, 1985. In Spanish.

[Sur89] R. Suri. Perturbation analysis: The state of the art and research issues explained via the GI/G/1 queue. *Proceedings of the IEEE*, 77(1):114–137, January 1989.

[SV77] G.N. Saridis and K.P. Valavanis. A hierarchical approach to the control of a prosthetic arm. *IEEE Transactions on Systems, Man and Cybernetics*, 7:407–420, 1977.

[SV88] G.N. Saridis and K.P. Valavanis. Analytical design of intelligent machines. *Automatica: the Journal of IFAC*, 24(2):123–133, 1988.

[SV90] M. Silva and R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets'89*, volume 266 of *LNCS*, pages 375–417. Springer-Verlag, 1990.

[Van78] H. Vantilborgh. Exact aggregation in exponential queuing networks. *Journal of the Association for Computing Machinery*, 25(4):620–629, October 1978.

[VZL87] M. Vernon, J. Zahorjan, and E. D. Lazowska. A comparison of performance Petri net models and queueing network models. In S. Fidia and G. Pujole, editors, *Modelling Techniques and Performance Evaluation*, pages 191–202. Elsevier Science Publishers B. V. (North Holland), 1987.

[WD92a] J. F. Watson, III and A. A. Desrochers. Methods for estimating state-space size of Petri nets. In *Proceedings of the 1992 IEEE Robotics and Automation Conference*, May 1992.

[WD92b] J. F. Watson, III and A. A. Desrochers. State-space size estimation of conservative Petri nets. In *Proceedings of the 1992 IEEE Robotics and Automation Conference*, May 1992.

[WDF85] C. Y. Wong, T. S. Dillon, and K. E. Forward. Timed place Petri nets with stochastic representation of place time. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 96–103, Torino, Italy, July 1985.

[WJG91] F. Wang, H. Jungnitz, and K. Gildea. Performance analysis of MMS using GSPN. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1573–1578, 1991.

[WL91] C.M. Woodside and Y. Li. Performance Petri net analysis of communications protocol software by delay-equivalent aggregation. Technical Report SCE-91-10, Telecommunications Research Institute of Ontario, Ontario, Canada, 1991.

[WS90] F. Wang and G. Saridis. A coordination theory for intelligent machines. *Automatica*, 26(5):833–844, 1990.

[Zub85] W. M. Zuberek. Performance evaluation using extended timed Petri nets. In *Proceedings of the 1985 Workshop on Timed Petri Nets*, pages 272–278, Torino, Italy, July 1985.