A Two-Level Decomposition Scheme For Markovian Process Algebra Models

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

durch den Fachbereich Wirtschaftswissenschaften,
Institut für Informatik und Wirtschaftsinformatik,
Universität Duisburg-Essen,
Campus Essen

Vorgelegt von
Name: Dipl.-Inf. Freimut Brenner
Geburtsort: Seoul, Korea
Essen, 2009

Tag der mündlichen Prüfung: 23.1.2009
Vorsitzender der Promotionskommission: Prof. Dr. Klaus Echtle
Erstgutachter: Prof. Dr. Bruno Müller-Clostermann
Zweitgutachter: Prof. Dr. Boudewijn R. Haverkort

Acknnowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. Foremost, I thank Prof. Dr. Bruno Müller-Clostermann for creating an ideal environment to develop and discuss new ideas. I am also indebted to Prof. Dr. Boudewijn R. Haverkort who made some valuable suggestions along the way. Finally, I thank Henrik Bohnenkamp who pointed out equation (IV.7).

Freimut Brenner
Essen, Februar, 2009

Abstract

The concurrent composition of Markov chains quickly leads to the notorious problem of state space explosion, also known as the largeness problem, as the number of involved Markov chains increases. In the context of Markovian Process Algebras (MPAs) this problem is of particular interest, since small and compact model descriptions in form of language terms provided by the MPA may possess huge underlying Markov chains. Of course, many long known counter-strategies to tackle the largeness problem of Markov chains in one way or the other, like, e.g., product-form solutions, lumpability, sparse data structures, nearly-complete decomposability, can also be applied to the Markov chains which are generated by MPA models. Recent research mostly focuses on the classification of syntactical properties on the MPA language term level which ensure the applicability of these strategies.

In this work we propose a novel approach to solve MPA models which explicitly exploits the concurrent nature of the given model. The method involves two levels of compositionality. In the first level, the model is decomposed along points of global synchronisation into several sub-models. These sub-models are solved in isolation, and afterwards the individual results are combined to yield a solution of the entire model. The second level of compositionality concerns the individual sub-models. Under certain conditions each sub-model can be described as the parallel evolution of a number of independent absorbing Markov chains. This independence can be exploited to efficiently solve the sub-models.

As a side result of the consideration of the second level of compositionality, we derive a novel result on cumulative measures of absorbing joint Markov chains. Provided that the marginal processes are independent continuous time Markov chains (CTMCs), the mean time to absorption and the expected total time in a transient set of the joint Markov chain are computed from the marginal CTMCs in a compositional way. Operations on the state space of the joint Markov chain are never carried out, hence, the problem of state space explosion is avoided. The computational effort of our method rather depends on convergence properties of the joint CTMC, i.e., the number of steps until absorption of a discrete time Markov chain embedded in the joint CTMC.

Zusammenfassung

Die nebenläufige Komposition von Markovketten führt mit steigender Anzahl an involvierten Komponenten schnell zum bekannten Problem der Zustandsraumexplosion, auch bekannt als Largeness Problem. Im Kontext von Markovschen Prozess-Algebren (MPA) ist dieses Problem von besonderer Bedeutung, da kleine und kompakte Modellbeschreibungen in Form von Sprachtermen riesige Markovketten repräsentieren können. Viele altbekannte Gegenstrategien zur Zustandsraumexplosion, wie z.B. Produkt-Form-Lösungen, Lumpability, dünnbesetzte Datenstrukturen, können auf die von der jeweiligen MPA erzeugten Markovkette angewendet werden. Die jüngste Forschung konzentriert sich vornehmlich auf die Klassifikation von syntaktischen Eigenschaften auf der MPA Sprachebene, welche die Anwendbarkeit dieser Strategien garantieren.

In der vorliegenden Arbeit schlagen wir einen neuen Ansatz zur Lösung von MPA Modellen vor, der explizit die nebenläufige Struktur des gegebenen Modells ausnutzt. Diese Methode besteht aus zwei Ebenen der Kompositionalität. In der ersten Ebene wird das Modell entlang von globalen Synchronisationspunkten in mehrere Submodelle aufgespalten. Diese Submodelle werden zunächst in Isolation gelöst; anschließend erhält man durch geeignete Kombination der einzelnen Lösungen eine Lösung für das gesamte Modell. Die zweite Ebene der Kompositionalität betrifft die individuellen Submodelle. Unter bestimmten Bedingungen kann jedes Submodell als die parallele Entwicklung mehrerer unabhängiger absorbierender Markovketten beschrieben werden. Diese Unabhängigkeit kann zur Lösung der Submodelle ausgenutzt werden.

Als ein Nebenprodukt der Betrachtung der zweiten Ebene der Kompositionalität, präsentieren wir ein neues Resultat über kumulative Maße gemeinsamer absorbierender Markovketten. Falls die marginalen Prozesse unabhängige kontinuierliche Markovketten sind, können die mittlere Zeit bis zur Absorption, sowie die mittlere Verweilzeit in einer transienten Teilmenge des Zustandsraums aus isolierten Lösungen der marginalen Prozesse zusammengesetzt werden. Da bei dieser Methode keine Operationen auf dem gemeinsamen Zustandsraum ausgeführt werden, umgehen wir das Problem der Zustandsraumexplosion. Der Rechenbedarf unserer Methode hängt von Konvergenzeigenschaften der gemeinsamen Markovkette ab, d.h. von der Anzahl an Schritten bis zur Absorption einer in der gemeinsamen kontinuierlichen Markovkette eingebetteten diskreten Markovkette.

Contents

I Introduction 1

II Markovian Process Algebras and Solution Techniques 5

II.1 MPA/PEPA . . . 6

II.2 MPAs and Equivalence Relations . . . 11

II.3 Harnessing Compositionality . . . 12

II.3.1 Product-Form Solutions . . . 13

II.3.2 Quasi-Separability . . . 14

II.3.3 Time-Scale Decomposition . . . 15

II.3.4 State Space Representation . . . 15

II.4 Conclusion . . . 16

III The Two-Level Decomposition Scheme for MPA Models 17

III.1 A General Decomposition Scheme for CTMCs . . . 18

III.1.1 Decomposition (intuitive description) . . . 18

III.1.2 Decomposition (mathematical background) . . . 20

III.1.3 An Illustrating Example. . . . 21

III.2 The Algorithm (Parts 1, 2 and 3) . . . 23

III.3 The Bohnenkamp Method . . . 25

III.4 The Algorithm: Part 1 – The Requirements . . . 26

III.5 The Algorithm: Part 2 – The Embedded DTMC . . . 27

III.5.1 Establishing the Embedded DTMC . . . 28

III.5.2 The Detailed Algorithm . . . 30

III.6 The Algorithm: Part 3 – The Subsystems . . . 31

III.6.1 Solving the Subsystems . . . 31

III.6.2 The Detailed Algorithm . . . 35
III.7 A Generic Example . . . 36
III.7.1 Example: The Embedded DTMC . . . 37
III.7.2 Example: The Subsystems . . . 40
III.8 An Application Example . . . 41
III.8.1 Model Description . . . 42
III.8.2 Preliminary Considerations . . . 43
III.8.3 Solving the Model *System* . . . 46
III.8.4 Comparison to the Bohnenkamp Method . . . 49
III.9 Conclusion . . . 51

IV Cumulative Measures of Absorbing Joint Markov Chains 55
IV.1 The Basic Setup . . . 56
IV.2 Well-Known Methods . . . 56
IV.2.1 Phase-Type Distributions . . . 56
IV.2.2 Numerical Integration . . . 57
IV.2.3 Uniformisation . . . 58
IV.2.4 MeanMax . . . 59
IV.3 Compositional Uniformisation . . . 61
IV.3.1 Probabilistic Interpretation . . . 63
IV.3.2 CU Algorithms and Complexity . . . 64
IV.4 Proofs of the Theorems 2 and 3 . . . 66
IV.4.1 Proof of Theorem 2 . . . 66
IV.4.2 Proof of Theorem 3 . . . 69
IV.5 CU: Two Generic Numerical Examples . . . 70
IV.5.1 Example Cases . . . 70
IV.5.2 Three Iterative Methods . . . 71
IV.5.3 Results . . . 73
IV.6 CU: A Special Case with an Exact Solution . . . 76
IV.7 Conclusion . . . 77

V Extensions of the 2-LDS 79
V.1 Terminating PEPA Processes: 2-LDS(T) . . . 79
V.1.1 Terminating Behaviour in PEPA . . . 80
V.1.2 Sketch of the 2-LDS(T) for Terminating PEPA Processes . . . 81
V.1.3 Example: A simple Producer-Consumer System with Failures . . . 86
V.2 Beyond Barrier Synchronisation: 2-LDS(P) . . . 91

V.2.1 Example: Sensor Network without Repair 92
V.2.2 Example: Extended Sensor Network with Repair 94
V.2.3 Sketch of the 2-LDS(P)-Algorithm 97
V.3 Conclusion 100

VI Conclusion **103**

Bibliography **105**

Appendices **109**

Chapter I

Introduction

The analysis of stochastic systems is a branch of applied research which spreads across many disciplines of science. Areas of application include performance and reliability of computing systems and telecommunication networks, and also fields like reaction kinetics in physical chemistry and financial risk theory, to name but a few.

Often a model of such a stochastic system is forced to fit into a Markovian framework, i.e., a Markov chain can be extracted from that model. As a consequence the model can be analysed by means provided by the rich and often elegant theory of Markov chains. In addition, formalisms to build (or describe) Markovian models exist. These formalisms constitute the advantage that they (may or may not) equip the model, or certain activities or states of that model, with an intuitive meaning. Queueing stations (and networks), stochastic Petri nets and Markovian Process Algebras are outstanding examples of formalisms which have shown their usefulness in the areas of performance and reliability evaluation of computer and communication networks. For an extended overview and application examples see, e.g., [7], [13], [18] and [11], or the latest proceedings of the conferences MMB ([2]), QEST ([17]) and SIGMETRICS ([35]).

When dealing with Markov chain formalisms, the first two questions should be: (1) Is that formalism useful from the modellers point of view? and (2) Can the special structure which the formalism induces on the underlying Markov chain be exploited to derive efficient (or elegant) solutions of that Markov chain? For instance, the solution of a birth-death process is given by a simple symbolic expression; BCMP networks possess a product-form solution ([1]); nearly completely decomposable Markov chains can be partitioned into sub chains, where transitions inside of sub chains and transitions between sub chains can be treated separately ([16]).

The current thesis mainly focuses on aspect (2), i.e., we exploit structure in Markov chains in order to derive solutions. The main parts of this thesis are:

(a) The task of the first main part, chapter III, is to contribute to the field of Markovian Process Algebras (MPAs). An MPA model consists of several concurrent components which may interact with each other through synchronisation. For a certain class of MPA models the components behave independently of each other when currently not involved in a synchronisation. This structure is exploited in order to access solutions of the MPA model.

(b) The second main part consists of chapter IV, where we present a new method (Compositional Uniformisation) to analyse cumulative measures of an absorbing Markov chain which is the joint process of several independent marginal absorbing CTMCs. Compositional Uniformisation calculates the moments of the time to absorption, as well as the expected total time which the joint CTMC spends in a certain transient subset of the state space.

We want to provide the reader with a more intense flair of these two aspects:

(a) Compositional Solution of MPAs. Stochastic process algebras have become popular since the formalism was proposed by Herzog in [23]. In particular, Markovian Process Algebras (MPAs) have drawn much attention due to the the fact that the quantitative solution of an MPA happens to be the solution of the underlying Markov chain. Examples for MPAs involve PEPA ([27]), EMPA ([3], [4]), MTIPP ([21]) and IMC ([20]).

Without going into the specifics of MPAs at this point we briefly sketch how the topic of absorbing joint Markov chains fits in the world of MPAs and how we plan to exploit it in chapter III. Consider the example sketched in Fig. I.1, where three processes Y_1, Y_2 and Y_3 evolve in parallel. As soon as all three processes are ready to synchronise, the synchronisation takes place (in some state $\in \{s, s', s'', \ldots\}$). After completion of the synchronisation the processes again start to evolve independently of each other until they are ready for the next synchronisation. By now you will most probably have realised that a sample path of this system is a concatenation of several situations as described in Fig. I.2. That means the entire system can be decomposed into several individual parts which themselves can be modelled by three independent absorbing Markov chains evolving in parallel. We anticipate that *solving* the entire model consists of two steps: 1. Solving these individual parts. 2. Determining how the individual parts connect to one another. This boils down to solving an embedded discrete time Markov chains with state space $\{s, s', s'', \ldots\}$. We'll get more specific on what *solving* in 1. and 2. actually means when delving into the details in chapter III.

(b) Cumulative Measures of Absorbing Joint Markov Chains. An important modelling pattern in many performance and reliability investigations are cyclic concurrent processes. Of particular interest are scenarios where a set of processes starts at the same instant of time, then proceed independently and finally synchronise in a shared event. We want to derive the time duration until all processes or a subset of these processes have

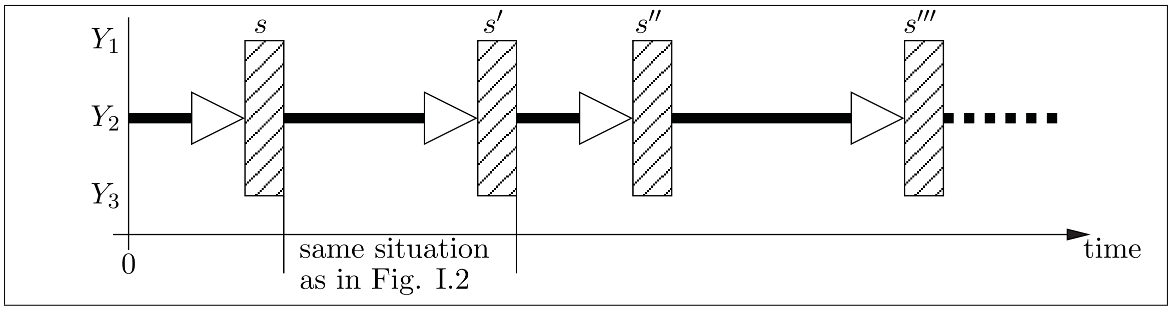


Figure I.1: MPA model with barrier synchronisations. The thick arrows are an aggregated representation of the independent evolution of the marginal processes Y_1, Y_2 and Y_3. The states in which synchronisations take place are denoted $s, s', s'', \ldots$.

finished their progress. Usually, in performance modelling the termination of a process means the completion of a task whereas in reliability modelling a process completion is associated with a component failure or the successful repair of a component.

The basic pattern of cyclic concurrent processes which posses a common global synchronisation point is sketched in Fig. I.2.

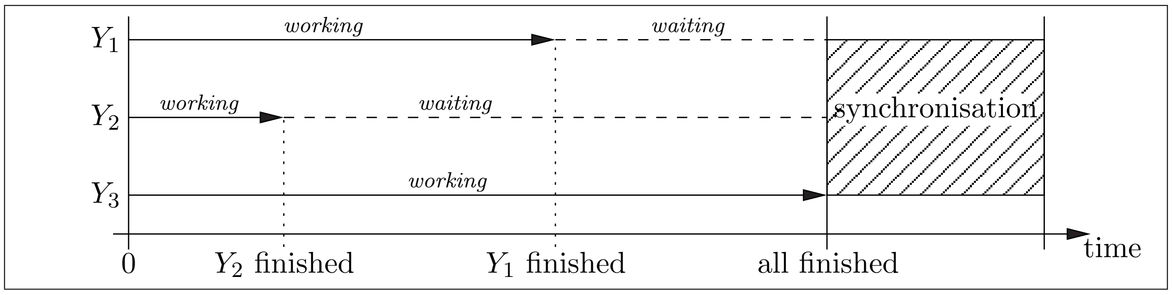


Figure I.2: Concurrent processes. Synchronisation begins when all processes have finished their work.

It is clear that the individual processes can be modelled by absorbing Markov chains. The time instant where the slowest of the chains becomes absorbed (i.e., the joint Markov chain becomes absorbed) models the time instant where all the processes have finished their work and the synchronisation begins.

This basic pattern occurs in many performance and reliability models. The measure of interest here is the time until all m processes do synchronise. Dependent on the application this measure may be also called response time, cycle time or time to absorption. In reliability evaluation one might be interested in the time until the process $m - k + 1$ out of m processes (or hardware devices) terminates which leads to the time to failure of a $k - of - m$ system.

Outline of the Thesis. In Chapter II a brief introduction into MPAs is given, as well as an (non-exhaustive) overview of solution techniques which tackle the problem of state space explosion inherent to MPA models. The particular MPA PEPA is introduced, since it will be used to illustrate our ideas throughout this work. The reader who is familiar with MPAs, and especially with PEPA, may well skip Chapter II.

In Chapter III we propose the two-level decomposition scheme (2-LDS) to solve a certain class of PEPA models for steady-state probabilities. This class contains all models which possess an ergodic underlying Markov chain and in which all sequential components must participate in every synchronisation, i.e., they perform barrier synchronisations. The latter restriction implies that every synchronisation can be mapped to a corresponding global state. The model under investigation can then be decomposed into subsystems and an embedded DTMC along the global synchronising states. We call this the first level of compositionality. The subsystems are either trivial, i.e., they consist of only one state, or they can be represented by the parallel composition of several independent absorbing CTMCs, which allows the application of efficient solution techniques. This is referred to as the second level of compositionality. The individual solutions of the subsystems combined with the solution of the embedded DTMC yields the solution of the original model.

In Chapter IV a novel result on cumulative measures of absorbing joint Markov chains is presented. Provided that the marginal processes are independent CTMCs, the mean time to absorption and the expected total time in a transient set of states of the joint Markov chain are computed from the marginal CTMCs in a compositional way. Operations on the state space of the joint Markov chain are never carried out, hence, the problem of state space explosion is avoided.

In Chapter V the two-level decomposition scheme is modified to be applicable to two new classes of MPA models. The first class are terminating PEPA processes with barrier synchronisations. Of course, for a terminating process only transient measures can be computed. The modified version of the 2-LDS which achieves this will be referred to as 2-LDS(T). The second class of MPA models describe systems in steady-state, but instead of barrier synchronisations the sequential components perform so called pre-emptive synchronisations. In a pre-emptive synchronisation a component which is ready to synchronise immediately initiates the (global) synchronisation, i.e., a component which is ready to synchronise never has to wait for the other participants to become ready on their part. The corresponding solution method is referred to as 2-LDS(P).

Finally, Chapter VI concludes this thesis.

Chapter II

Markovian Process Algebras and Solution Techniques

In this chapter we highlight a few aspects of Markovian Process Algebras (MPAs) with the intention of providing the reader with the necessary background for the remainder of this work. To gain an exhaustive understanding of MPAs we refer to publications concerned with the concrete MPA languages PEPA ([27]), EMPA ([3], [4]), MTIPP ([21]) and IMC ([20]). MPAs are a special case of Stochastic Process Algebras (SPAs) which were proposed by Herzog in [23]. In short, an SPA model consists of a set of language terms, where each term describes a process which may be characterised by delays and choices. Delays are stochastic, i.e., they are drawn from a certain probability distribution, and choices may be deterministic, non-deterministic or stochastic. In addition, several processes, language terms respectively, can be combined via a Prefix-operator (i.e., serial execution of the processes), a choice operator or a concurrency operator, such that complex systems can be modelled in a compositional and hierarchical way. SPAs allow functional aspects (deadlock-freeness, reachability properties), as well as quantitative aspects (throughputs, sojourn times) to be included in a single model.

Solving an SPA model for quantitative properties consists of two steps: (1) Find, i.e., construct, the stochastic process which is described by the given SPA language term. (2) Solve this stochastic process. The stochastic process underlying an SPA model usually consists of a finite number of states, and the sojourn times (delays) in the individual states may be quite general. The solution of such a general stochastic process comprises the solution of a set of differential equations, or even worse, depending on the type of process. If, however, the underlying stochastic process happens to be an ergodic[1] continuous time Markov chain the computation of steady-state probabilities simplifies to solving a set

[1]A CTMC is said to be ergodic, iff it is irreducible and positive recurrent. If the CTMC is finite, i.e., its state space is finite, irreducibility implies positive recurrence.

of linear equations. Forcing all delays in an SPA model to be drawn from exponential distributions guarantees that the underlying stochastic process is a Markov process. An SPA which imposes this condition on delays is said to be a Markovian Process Algebra (MPA).

The most frequent problem which arises when attempting to solve an MPA model is that the underlying Markov chain is subject to state space explosion. This is due to the fact that an MPA model can be the concurrent composition of several processes (language terms). If, e.g., we have m processes which are combined via concurrent composition, where each process consists of at most K states, then the composite model consists of at most K^m states.

In section II.1 we give a short introduction in the MPA PEPA (Performance Evaluation Process Algebra) which was introduced by Hillston in [27]. Although, a couple of other MPAs do exist, we will use this specific MPA to illustrate our ideas. For the sake of completeness section II.2 makes a brief comment on the role of equivalence relations in MPAs. Finally, section II.3 gives a selection of a few solution methods for MPAs which all deal with the problem of state space explosion. We restrict this selection to methods which are well-known in the general context of Markov chains (i.e., product-form solutions, nearly complete decomposability, efficient state space representation) and adjusted to fit into the specific framework of MPAs.

II.1 MPA/PEPA

Definition 1. The language defined by the following grammar is the set of all possible PEPA expressions.

$$C \ := \ (\alpha, r).C \mid C + D \mid C/L \mid C \underset{L}{\bowtie} D \mid M. \tag{II.1}$$

$\square$

Each expression of the language defined in Def. 1 describes a process whose behaviour is determined by the following rules:

Prefix: The process $(\alpha, r).C$ executes the activity (α, r) – which possesses the action type α and an exponentially distributed duration with rate r, $r \in \mathbb{R}_{>0}$ – and afterwards behaves like C. It is also possible to leave the rate r unspecified, in which case we use the symbol $\top$.

Choice: In a process $C + D$ all currently enabled activities in C and D are involved in a race condition. The activity to win this race is executed. Due to the memoryless property of the exponential distribution all other activities are reset. If for example in the process $(\alpha, r).C + (\beta, v).D$ the activity (α, r) wins, then afterwards the process behaves like $C + (\beta, v).D$.

We prelude the description of cooperation, or synchronisation of PEPA components with the following definition.

Definition 2. The apparent rate $r_\alpha(C)$ of an activity of action type α in component C is defined by

$$\begin{aligned}
r_\alpha((\beta, r).C) &= \begin{cases} r, & \text{if } \alpha = \beta \\ 0, & \text{if } \alpha \neq \beta \end{cases} \\
r_\alpha(C/L) &= \begin{cases} r_\alpha(C), & \text{if } \alpha \notin L \\ 0, & \text{if } \alpha \in L \end{cases} \\
r_\alpha(C + D) &= r_\alpha(C) + r_\alpha(D) \\
r_\alpha(C \underset{L}{\bowtie} D) &= \begin{cases} r_\alpha(C) + r_\alpha(D), & \text{if } \alpha \notin L \\ \min\{r_\alpha(C), r_\alpha(D)\}, & \text{if } \alpha \in L \end{cases}
\end{aligned}$$

□

The apparent rate $r_\alpha(C)$ returns the sum of the rates of all activities of type α which are currently enabled in C. In other words, $r_\alpha(C)$ is the overall rate at which activities of type α are currently observed in C.

Cooperation: $C \underset{L}{\bowtie} D$ denotes the situation where the components C and D must synchronise over activities which are of an action type contained in the synchronisation set L. Activities of this kind are called shared activities. C and D evolve independently of each other (i.e., in parallel) until the first of the two components, say C, *reaches* a shared activity. From this time instant on this shared activity becomes blocked in C until also D reaches a shared activity of the same action type. If this happens the shared activity is executed simultaneously by C and D. If C and D synchronise over a shared activity with type α, then the rate of the shared activity is the minimum of the apparent rates $r_\alpha(C)$ and $r_\alpha(D)$. The semantics of this is that each component which actively participates in a synchronisation must complete some work before interaction is achieved, i.e., before the synchronisation finishes, hence, the rate of synchronisation is limited by the rate of the slowest participant. Other MPAs employ different semantics for the synchronisation mechanism, which results in different choosings of the rate at which synchronisation proceeds; for a more thorough treatment see [26]. If one or more activities involved in the synchronisation possess an unspecified rate, then these activities can be regarded as passive – they are not taken into account when determining the rate of the shared activity.

Hiding: C/L has the meaning that the action type of all activities in C which are of an action type contained in L are hidden to the *outside* of C. Hidden activities are not executed in cooperation with other components. Nevertheless, inside of component C these actions are still visible.

Constant: Constants are components whose meaning is given by a defining equation. For two constants M and E, $M \overset{def}{=} E$ assigns M the behaviour of component E.

The semantics for the PEPA language is given by the structured operational semantics rules (SOS-rules) in figure II.1. A rule of the form $\frac{B}{C}A$ is read as: Given A, B implies C. If A is missing, then there is no precondition. If B is missing, then C holds, provided A.

Definition 3. Let C be a PEPA process.

- The one-step derivative set $ds^{(1)}(C)$ of C contains all the processes C' which can be reached by applying the SOS-rules once to C.
- The k-step derivative set $ds^{(k)}(C)$ of C, $k \geq 2$, is given by $ds^{(k)}(C) = \bigcup_{C' \in ds^{(k-1)}(C)} ds^{(1)}(C')$.
- The derivative set $ds(C)$ of C is given by $ds(C) = \bigcup_{k=1}^{\infty} ds^{(k)}(C)$.

□

Starting with a process C the *derivation graph* can be constructed by successive application of the SOS-rules. The elements of the derivative set $ds(C)$ form the nodes of this graph. There exists an edge from node C_1 to C_2, $C_1, C_2 \in ds(C)$, iff $C_2 \in ds^{(1)}(C_1)$, i.e., iff there exists an activity (α, r) which causes C_1 to evolve into C_2. Edges are labelled by the corresponding activities. Note that the derivation graph might be a multi graph since C_1 may evolve into C_2 through different activities.

A CTMC is obtained from the derivation graph by

1. considering nodes as states
2. abstraction from action types within activities
3. amalgamation of multi edges into a single edge, where the activity rates are summed up.

Definition 4. A PEPA component C is said to be cyclic, or irreducible, if $C \in ds(C')$ for all $C' \in ds(C)$.

□

The importance of cyclic PEPA components arises from the fact that the CTMC associated with this component is irreducible if and only if the PEPA component is cyclic. Since PEPA

Prefix :

$$\frac{}{(\alpha, r).C \xrightarrow{(\alpha,r)} C}$$

Choice :

$$\frac{C \xrightarrow{(\alpha,r)} C'}{C + D \xrightarrow{(\alpha,r)} C'} \qquad \frac{D \xrightarrow{(\alpha,r)} D'}{C + D \xrightarrow{(\alpha,r)} D'}$$

Cooperation :

$$\frac{C \xrightarrow{(\alpha,r)} C'}{C \underset{L}{\bowtie} D \xrightarrow{(\alpha,r)} C' \underset{L}{\bowtie} D}(\alpha \notin L) \qquad \frac{D \xrightarrow{(\alpha,r)} D'}{C \underset{L}{\bowtie} D \xrightarrow{(\alpha,r)} C \underset{L}{\bowtie} D'}(\alpha \notin L)$$

$$\frac{C \xrightarrow{(\alpha,r_1)} C' \quad D \xrightarrow{(\alpha,r_2)} D'}{C \underset{L}{\bowtie} D \xrightarrow{(\alpha,R)} C' \underset{L}{\bowtie} D'}(\alpha \in L) \qquad \text{where} \quad R = \frac{r_1}{r_\alpha(C)} \frac{r_2}{r_\alpha(D)} \min(r_\alpha(C), r_\alpha(D))$$

Hiding :

$$\frac{C \xrightarrow{(\alpha,r)} C'}{C/L \xrightarrow{(\alpha,r)} C'/L}(\alpha \notin L) \qquad \frac{C \xrightarrow{(\alpha,r)} P'}{C/L \xrightarrow{(\tau,r)} C'/L}(\alpha \in L)$$

Constant :

$$\frac{C \xrightarrow{(\alpha,r)} C'}{D \xrightarrow{(\alpha,r)} C'}(D \overset{def}{=} C)$$

Figure II.1: SOS-rules of PEPA

components define only finite state CTMCs, irreducibility implies positive recurrence of that CTMC.

In [27] it is shown that a necessary condition for a PEPA component to be cyclic is that all choices must occur within cooperating PEPA components. Thus, every cyclic PEPA component can be constructed out of the following grammar.

$$\text{sequential components} \quad R \; := \; (\alpha, r).R \mid R + R \mid M \tag{II.2}$$

$$\text{model components} \quad C \; := \; R \mid C/L \mid C \underset{L}{\bowtie} C. \tag{II.3}$$

Example: Constructing the Underlying CTMC Consider the following simple Producer-Consumer model, where synchronisation proceeds over the action label *handover*.

$$\begin{aligned} \mathit{Producer} &= F_0 & \mathit{Consumer} &= G_0 \\ F_0 &= (\mathit{produce}, p).F_1 & G_0 &= (\mathit{handover}, h).G_1 \\ F_1 &= (\mathit{handover}, h).F_0 & G_1 &= (\mathit{consume}, c).G_0 \end{aligned}$$

Fig. II.2 shows the labelled transition systems of the components *Producer* and *Consumer*. In Fig. II.3 the derivation graph of the composite component $\mathit{Producer} \underset{\{handover\}}{\bowtie} \mathit{Consumer}$, as well as the CTMC underlying the composite component are shown, where state (i, j) of the CTMC corresponds to the component $F_i \underset{\{handover\}}{\bowtie} G_j$.

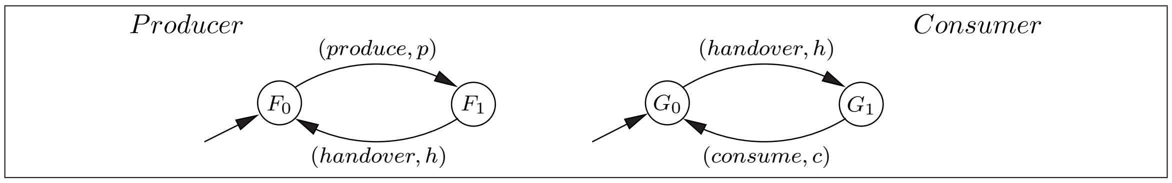


Figure II.2: The components *Producer* and *Consumer*.

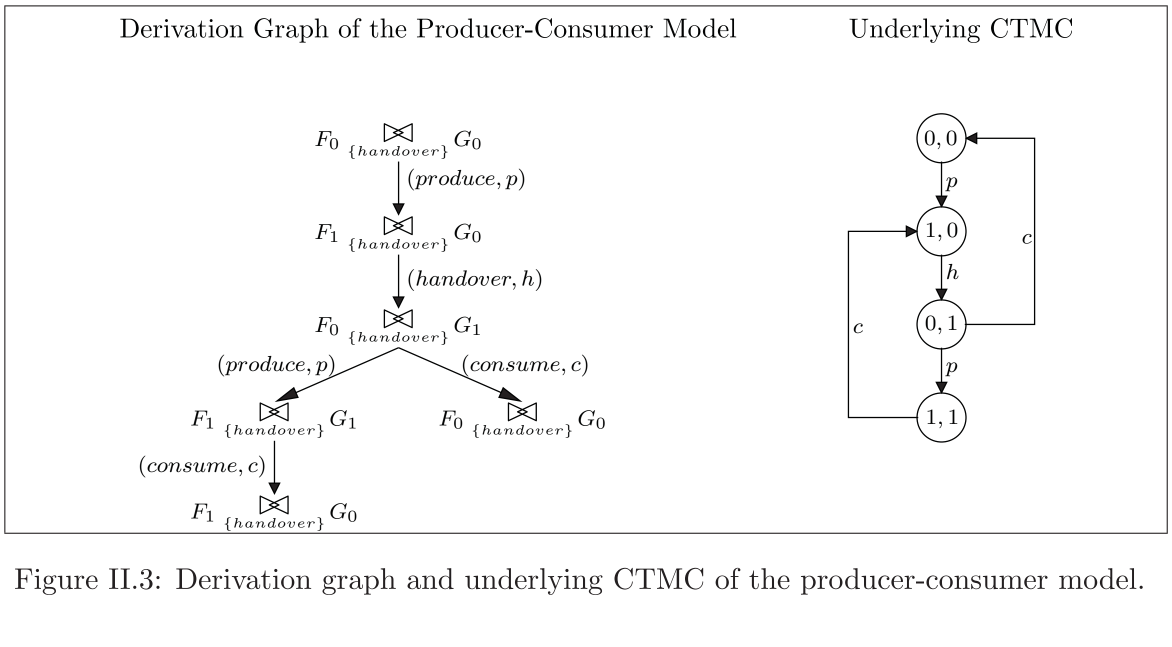


Figure II.3: Derivation graph and underlying CTMC of the producer-consumer model.

Notational Conventions

components $\leftrightarrow$ states. Throughout this chapter we will often use the terms component and state interchangeably. Consider the above Producer-Consumer Model: If, e.g., we speak of the component $F_1 \underset{L}{\bowtie} G_1$ we sometimes refer to the PEPA component or to the state $(1, 1)$ of the underlying Markov chain. Conversely, if we speak of the state $(1, 1)$ we sometimes refer to the state in the Markov chain or the corresponding PEPA component $F_1 \underset{L}{\bowtie} G_1$.

Applying the cooperation operator. Whenever we speak of *applying the cooperation operator* to several components $C_1, \dots, C_m$, we mean that the one-step derivatives of $C := C_1 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} C_m$ are determined, as well as the rates of all activities enabled in that component. If the component C corresponds to a state $\mathbf{x} = (x_1, \dots, x_m)$ in the underlying Markov chain, we may also speak of applying the cooperation operator to $x_1, \dots, x_m$.

Global and local states. When considering PEPA components and their underlying Markov chains we often have to distinguish between local states and global states. We will always represent global states by small bold letters and local states by small normal letters. E.g. the state of the CTMC underlying the component $C = C_1 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie}$ may be represented by $\mathbf{x} = (x_1, \dots, x_m)$.

II.2 MPAs and Equivalence Relations

Usually MPAs come equipped with at least one notion of equivalence. That means for a given equivalence relation $\equiv$, two components C and D are said to be equivalent $(C \equiv D)$ iff from a certain point of view they are the same. The existence of such an equivalence relation allows to investigate the component D instead of C, which is particularly useful if D is easier to solve (e.g., the underlying Markov chain possesses less states) than component C. If the equivalence relation $\equiv$ is preserved by the operators of the MPA, then $\equiv$ is a congruence relation. If, e.g., $C \equiv D$ and $C \underset{L}{\bowtie} F$ imply $(C \underset{L}{\bowtie} F) \equiv (D \underset{L}{\bowtie} F)$, then $\equiv$ is a congruence with respect to the cooperation operator. Provided that $\equiv$ is a congruence with respect to all given operators of the given MPA, a complex composite component can be simplified by replacing sub-components (i.e., building blocks of the composite component) hierarchically by their equivalent counterparts. Only after that substitution procedure does the Markov chain underlying the composite component have to be generated and solved.

In PEPA there exists the notion of strong equivalence. This equivalence relation manifests itself in lumpable partitions of the state space on the Markov chain level. As a consequence, a PEPA model can be replaced by a strongly equivalent model such that the Markov chain underlying the new model is a lumped version of the Markov chain underlying the original model. Hence, strong equivalence is a powerful tool to reduce the states of the Markov chain underlying a PEPA component by exact aggregation.

In addition, it can be shown that strong equivalence is a congruence in PEPA. In a composite PEPA model components can be replaced by equivalent components in a hierarchical manner, such that the resulting composite model is equivalent to the original composite model.

We give a short non-technical description of what strong equivalence is. First, the conditional rate $q(C, D, \alpha)$ from component C to component D is the rate at which component C evolves into component D via activities with action label α. The total conditional rate from component C into a set of components S via the rate α is given by $q(C, S, \alpha) := \sum_{D \in S} q(C, D, \alpha)$. Since strong equivalence is an equivalence relation it establishes a set of equivalence classes. Two PEPA components C and D are said to be strongly equivalent if, for any action type α and any equivalence class S we have $q(C, S, \alpha) = q(D, S, \alpha)$. That means the conditional rates from C and D into the same equivalence class are the same.

II.3 Harnessing Compositionality

Whenever parallelism or concurrency is introduced between a number of Markovian models this soon leads to the problem of state space explosion, also called the largeness problem. Consider a (global) model consisting of m local concurrent models, where the i-th local model has a state space of size K_i. Since every combination of local states might represent a global state, the state space of the global model can potentially possess $\prod_{i=1}^{m} K_i$ states. We say potentially because depending on the actual models under consideration not every combination necessarily represents a valid global state.

From the modeller's point of view one of the main benefits of MPAs is the possibility to specify complex concurrent systems by combining several possibly small (local) components via the cooperation operator. That means a system with a huge underlying Markov chain can be specified without explicitly specifying the Markov chain. This benefit usually does not carry through when performance measures of the system have to be derived. To obtain these measures it is necessary to solve the underlying Markov chain, i.e., to compute the steady-state distribution or transient distributions of the Markov chain. The compositional character of the system on the MPA level, though, often cannot be exploited to solve the Markov chain in a compositional way.

There do, however, exist a few promising approaches that try to tackle the largeness problem. Most of these approaches are well-known methods applicable to Markov chains in general, and some are especially tailored to the specifics of MPAs. In the following we outline a few of the approaches which are well-known from a general Markov chain context. For an exhaustive overview see [28] and [29].

II.3.1 Product-Form Solutions

One of the most desirable properties a global Markovian model can exhibit is that of a product-form solution. If a product-form solution exists, the local models can be solved in isolation for their local steady-state distributions, and afterwards these distributions can be combined via a tensor expression to yield the steady-state distribution of the global model. For example, in the composite PEPA component $C_1||C_2||\cdots||C_m$ the local components are arranged in parallel, i.e., they act completely independently of each other. Hence, the steady-state distribution can be expressed as

$$\pi_1 \otimes \pi_2 \otimes \cdots \otimes \pi_m,$$

where π_i is the steady-state distribution of the component C_i in isolation and $\otimes$ denotes the tensor product operator. Of course, the more interesting question is whether a product-form solution also exists if the local components actually do cooperate with each other via synchronisations.

Reversibility and Quasi-Reversibility Two properties which ensure a product-form solution are reversibility and quasi-reversibility of the global Markov chain. An ergodic Markov chain, with steady-state distribution $\pi(\cdot)$, is said to be reversible if any two states $i \neq j$ satisfy the detailed balance equations:

$$\pi(i)q(i,j) = \pi(j)q(j,i),$$

where $q(i,j)$ is the transition rate from state i to state j. That means, in equilibrium the flow from i to j equals the flow from j to i.

An ergodic Markov chain is said to be quasi-reversible if the partial balance equations are satisfied for all states i and corresponding subsets of the state space E':

$$\pi(i)\sum_{j\in E'} q(i,j) = \sum_{j\in E'} \pi(j)q(j,i).$$

That means, in equilibrium the flow from i to E' equals the flow from E' to i.

In [25] a class of PEPA components is identified which possesses reversible underlying Markov chains. It is further investigated under which conditions the property of reversibility is preserved by the PEPA combinators. A similar line of argumentation concerning quasi-reversible PEPA components is presented in [19].

Other Product-Forms In [24] and [31] the authors investigate a class of SPA models, where although components do interact via cooperation the global component still possesses a product-form solution. The considered scenario is characterised by several components which do not interact directly with each other but compete over a resource.

Once a component is in possession of the resource, all other competing components become blocked until the release of the resource. If, e.g., the components $C_1, C_2, \ldots, C_N$ compete over a resource B, the corresponding PEPA construct looks like

$$(C_1||C_2||\cdots||C_N) \underset{S}{\bowtie} B, \tag{II.4}$$

where S is some set of synchronising actions. It can be shown that the steady-state distribution of the composite component can be expressed as $c\otimes_i \pi_i$, where π_i is the steady-state distribution of $C_i \underset{S}{\bowtie} B$, and c is a normalising constant. That means, to obtain the product-form solution, the components $C_i \underset{S}{\bowtie} B$ are solved in isolation and afterwards the solutions are combined with a suitable normalisation constant. Computation of the normalisation constant, though, is the bottleneck of this approach. This approach is an extension of the work of Boucherie in [8], where the same form of interaction via a resource object is considered – the constituent competing processes, though, are not SPA components but pure Markov processes.

II.3.2 Quasi-Separability

Consider a Markovian system consisting of N sub-models, where the state of sub-model i can be described by the pair of variables (x_i, y_i). Then the state of the entire system can be described by the two vectors $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{y} = (y_1, \ldots, y_N)$. If consideration of the pair (x_i, y_i) is sufficient to completely describe the behaviour of model i, the sub-models act independent of each other – the system is said to be separable, and a product-form solution exists. If for the analysis of sub-model i the pair $(x_i, \mathbf{y})$ or $(y_i, \mathbf{x})$ must be considered, then the system is said to be quasi-separable. Say, the state of sub-model i is unambiguously described by $(x_i, \mathbf{y})$, then the sub-model i can be solved in isolation by considering state changes of $(x_i, \mathbf{y})$. Although, the individual solutions cannot necessarily be combined to yield a product-form solution of the entire system, at least local performance measures can be calculated.

In [43] quasi-separability is applied to SPA models. For the sub-model i the authors introduce a SPA component C_i which captures the dynamics that corresponds to state changes of the element x_i. These SPA components act independently of each other. In addition, a scheduler component B is defined which captures the dynamics of the entire system corresponding to state changes of $\mathbf{y}$. Then, the SPA model looks like

$$(C_1||C_2||\cdots||C_N) \underset{S}{\bowtie} B. \tag{II.5}$$

The sub-model i can then be analysed by considering the component $C_i \underset{S}{\bowtie} B$ in isolation. Although, this procedure looks similar to the procedure described in [24] and [31] (compare equation (II.5) to (II.4)), the difference lies in the fact that here the component B has much more complex possibilities of interaction with the components C_i, $i = 1 \cdots N$. This in general does not give rise to a product-form solution, as was the case with (II.4).

II.3.3 Time-Scale Decomposition

If the state space of a Markov chain can be partitioned such that the transition rates between states of the same partition outweigh the transition rates between states of different partitions by several orders of magnitude, the Markov chain is said to be nearly completely decomposable ([16]). The evolution of the Markov chain inside the partitions proceeds faster than the evolution between partitions – the Markov chain can be thought of as evolving on different time scales. Then the states can be ordered such that the generator matrix possesses a block diagonal structure, where the diagonal blocks contain the great transition rates (transitions inside a partition) and the off-diagonal blocks contain the small transition rates (transitions between partitions). The solution of the Markov chain consists of two steps: (1) Capture the dynamics inside the partitions by treating the diagonal blocks in isolation; (2) Capture the dynamics between partitions by establishing a Markov chain, in which each state corresponds to a partition of the original Markov chain.

In [30] a class of MPA models is identified which are likely to generate nearly completely decomposable Markov chains. These models consist of sequential components each of which possesses either only fast or only slow transitions. Mertsiotakis extends this work in [36] and [37] by also considering sequential components which may possess both fast and slow transitions.

II.3.4 State Space Representation

The simplest way to gain access to the elements of the generator matrix of the Markov chain underlying a composite MPA model is to explicitly construct this matrix and store it in the memory of the computer where computations over the Markov chain are to be carried out. Due to the possible largeness problem of the underlying Markov chain, the generator may become too big to be stored in memory. In this case it may be beneficial to employ an implicit representation of the generator matrix. That means, certain information about the Markov chain is stored, from which the generator elements can quickly be calculated if needed. We briefly describe two methods for an implicit representation of the generator matrix.

Kronecker Algebra In this approach, at first the generator matrices of the local MPA components are considered. Then a suitable Kronecker algebra which includes the Kronecker sum and the Kronecker product is defined. Access to a certain element of the generator matrix of the composite component is gained by applying the Kronecker algebra to the local generator matrices. During this process, the global generator matrix is never constructed – rather the individual matrix elements are generated on the fly. In [41] Plateau et al. proposed to apply Kronecker algebra to stochastic automata network. In [12] Buchholz investigates an MPA (MPA is also the name of that Markovian Process Algebra) to which the concept of representing the global dynamics as tensor expressions is inherent. In [42] the desire to use tensor algebra for MPAs results in a modified version of the MPA TIPP.

Ordered Binary Decision Diagrams (OBDDs) An effective data structure to store the generator matrix of a large Markov chain is the multi-valued ordered binary decision diagram. Binary decision diagrams (BDDs) can be used to represent boolean functions. A BDD is an acyclic directed graph, where except for the terminal nodes each node possesses two child nodes. The edges to the child nodes are labelled with 1 or 0. In addition each terminal node possesses either the value 1 or 0. A path from the root to one of the terminal nodes (i.e., a sequence of ones and zeroes) can be interpreted as the input of the boolean function, and the value of the corresponding terminal node is the value of the function. If in each possible path the order of the boolean variables is preserved, then we have an ordered BDD (OBDD). Usually, one is interested in a reduced OBDD, where isomorphic subgraphs are aggregated or merged into a single subgraph. If a decision node is allowed to possess more than two terminals which in addition can be assigned different values other than true or false (e.g., real numbers), then we speak of a multi-valued or multi-terminal OBDD, or multi-terminal binary decision diagram (MTBDD).

For a Markov chain a suitable multi-valued OBDD can be constructed, such that each possible path from the root to one of the terminals represents a transition of the Markov chain. That means, each transition is encoded as a sequence of zeroes and ones. The terminal of such a path contains just the rate of the encoded transition. For a general overview see, e.g., [22], [38] or [15], and especially the PhD thesis of Kuntz [33] or Lampka [34].

II.4 Conclusion

In this chapter we abridged a few aspects of Markovian Process Algebras, where the focus was laid on the MPA PEPA. The intention is to provide the reader with the necessary specifics of PEPA which will be presupposed in the remainder of this work. Some solution techniques which tackle the state space explosion problem inherent to MPA models were briefly discussed, where the emphasis was put on techniques which are well established in the general context of Markov chains and were transferred to fit into the MPA framework. For an exhaustive overview see [28] and [29]. Another solution technique which was proposed by Bohnenkamp in [5] will be outlined in Section III.3, since it will be the starting point of our own approach to solving MPA models.

Chapter III

The Two-Level Decomposition Scheme for MPA Models

In this chapter, we propose an algorithm to compute steady-state probabilities of a composite PEPA component in a compositional way. For this reason, the CTMC underlying the PEPA component is decomposed into several subsystems which can be solved in isolation. After that, the individual solutions (of the subsystems) are combined into the desired steady-state probability of the entire CTMC via a well-known result from the theory of semi-regenerative processes. Our algorithm is an extension of Bohnenkamp's work [5], where he was able to solve a semi-Markov chain, which is embedded in the considered CTMC, in a similar fashion. Preliminary considerations which lead to the results of the current chapter were already published in [10].

Section III.1 explains a decomposition scheme that can generally be applied to CTMCs. Based on this decomposition scheme the basic structure of our algorithm which consists of three parts is introduced in section III.2. Similarities and differences between our approach and Bohnenkamp's method are highlighted in section III.3. The three parts of our algorithm are discussed in detail in the sections III.4, III.5 and III.6. An application example of our algorithm to a generic PEPA model is given in section III.7, where the main purpose is the illustration of all steps which our algorithm runs through. Another example which is more in step with actual practise is given in section III.8. It is directly borrowed from [5], with minor modifications which are due to the translation of the model from the MPA $\mathscr{YAWN}$ to PEPA.

III.1 A General Decomposition Scheme for CTMCs

This section explains the general decomposition scheme which we later want to apply to composite PEPA components. This decomposition scheme exploits the semi-regenerative property of continuous time Markov chains, hence, it can be applied to every CTMC. For better readability, we first describe the decomposition in an intuitive way, and justify it mathematically afterwards.

III.1.1 Decomposition (intuitive description)

Let $Z = (Z_t)_{t\geq 0}$ be a continuous time Markov chain with state space E, and let $A \subset E$. Our target quantity is the steady-state probability $\mathbb{P}(Z \in A)$ that Z is in the set A. Let $E^{(X)} \subset E$ be an arbitrary subset of the state space. The set of embedded states $E^{(X)}$ defines an embedded DTMC X, hence, the notation $E^{(X)}$. Then we can think of Z as being composed out of several subsystems as follows: If Z enters a state $x \in E^{(X)}$, we say that Z starts to behave as subsystem $SUB(x)$. As soon as Z enters another state $x' \in E^{(X)}$, Z starts to behave as subsystem $SUB(x')$. That means, $SUB(x), x \in E^{(X)}$, is actually a stochastic time-dependent process.

Definition 5. Let T_1 be the time instant at which Z visits the set $E^{(X)}$ for the first time, where it does not count as a visit if $Z_0 \in E^{(X)}$, i.e., if Z is initially in $E^{(X)}$. Then, the stochastic process $SUB(x) = (SUB(x)_t)_{t\geq 0}$, $x \in E^{(X)}$, is given by

$$\mathbb{P}(SUB(x)_t = y) = \mathbb{P}(Z_t = y \wedge T_1 > t | Z_0 = x), \text{ for all } y \in E.$$

□

Fig. III.1 illustrates how Z adopts the behaviour of different subsystems after visiting states contained in $E^{(X)}$.

Remark 1. In the remainder of this work, we will sometimes speak of Z as entering (or visiting) the subsystem $SUB(x)$, $x \in E^{(X)}$. By this, we mean that Z visits the state $x \in E^{(X)}$ and starts to behave as $SUB(x)$. Analogously, if we use the term of Z leaving a subsystem $SUB(x)$ at a time instant t, we mean that Z ceases to behave as $SUB(x)$ because $T_1 > t$. Upon leaving a subsystem $SUB(x)$, $x \in E^{(X)}$, Z invariably enters some subsystem $SUB(x')$, $x' \in E^{(X)}$, where the case $x = x'$ is possible. We can think of Z as being routed to a subsequent subsystem upon leaving a subsystem. We call this the dynamics between subsystems.

The steady-state probability $\mathbb{P}(Z \in A)$ is the relative amount of time which a random subsystem spends in the set A. By *random* we mean that a subsystem is chosen randomly according to its relative visiting frequency within the original process Z. For a fixed $x \in E^{(X)}$, $SUB(x)$ is a fixed subsystem, and it becomes a random subsystem if $x \in E^{(X)}$ is chosen randomly.

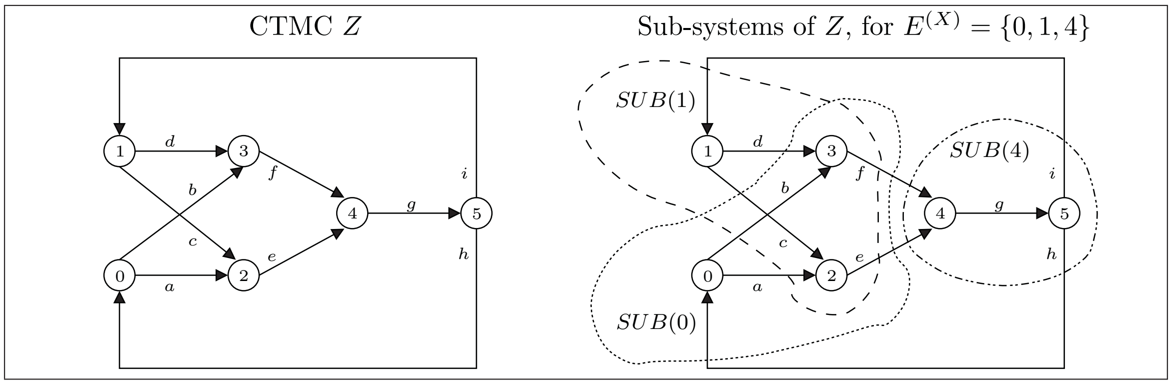


Figure III.1: Decomposition of an example CTMC Z into subsystems. $E^{(X)}$ is a given subset of the state space of Z. Each state of $E^{(X)}$ defines the entry point of a subsystem. The labels $a, b, \ldots, i$ denote transition rates.

For a fixed $x \in E^{(X)}$, be $T(x)$ the mean sojourn time of $SUB(x)$, $T(x)(A)$ the expected total time which $SUB(x)$ spends in A, and $\pi(x)$ the relative visiting frequency of $SUB(x)$. Then, the following is obvious:

$$\sum_{x \in E^{(X)}} \pi(x)T(x) : \text{mean sojourn time of a random subsystem,}$$

$$\sum_{x \in E^{(X)}} \pi(x)T(x)(A) : \text{expected total time in } A \text{ of a random subsystem.}$$

As a consequence, the relative amount of time a random subsystem spends in A, i.e., the steady-state probability $\mathbb{P}(Z \in A)$, is given by

$$\mathbb{P}(Z \in A) = \frac{\sum_{x \in E^{(X)}} \pi(x)T(x)(A)}{\sum_{x \in E^{(X)}} \pi(x)T(x)}. \tag{III.1}$$

The decomposition of Z into subsystems $SUB(x)$, $x \in E^{(X)}$, defines a DTMC with state space $E^{(X)}$ which is embedded in Z. This embedded DTMC describes the dynamics between the subsystems. More specifically, a transition probability $\mathbb{P}(x \to x')$ of the embedded DTMC equals the routing probability from subsystem $SUB(x)$ to the subsystem $SUB(x')$. As a consequence, the visit frequency $\pi(x)$ of subsystem $SUB(x)$ is given by the steady-state probability of the state x of the embedded DTMC.

An overview graphic which visualises the decomposition of a CTMC Z into subsystems and an embedded DTMC X is given in Fig. III.2.

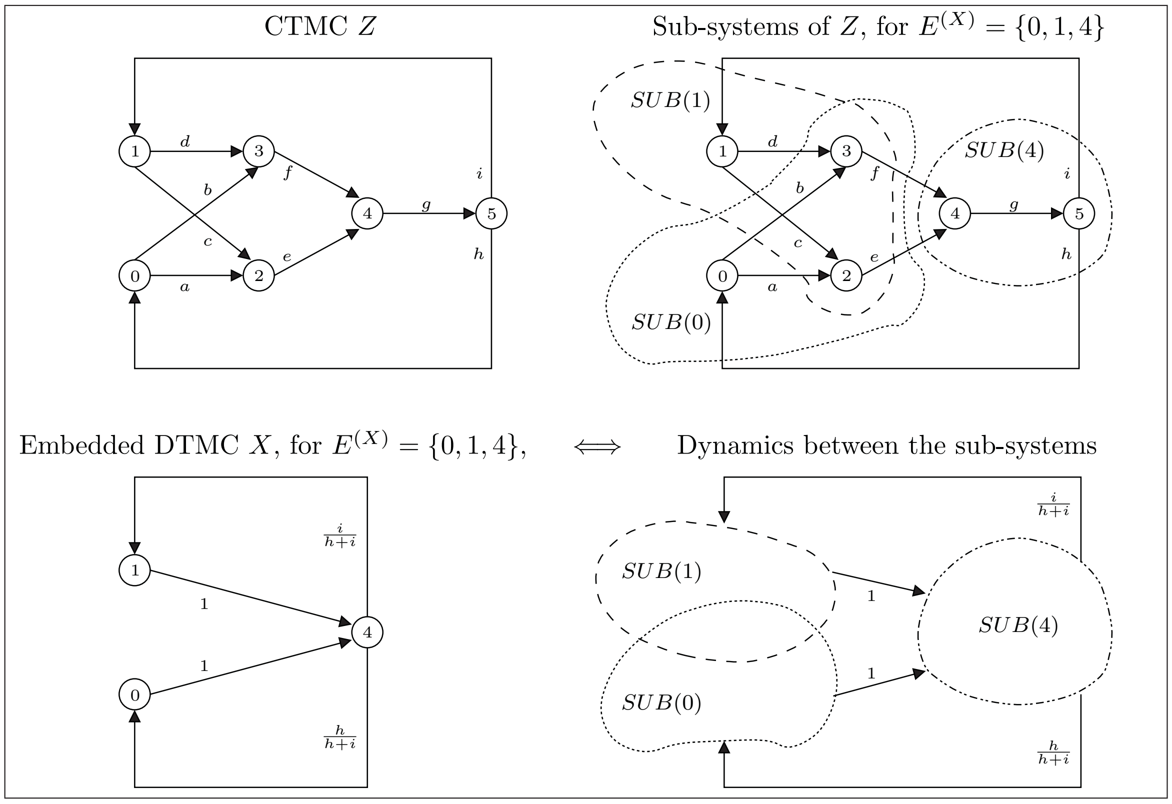

CTMC Z
Sub-systems of Z, for $E^{(X)} = \{0,1,4\}$
Embedded DTMC X, for $E^{(X)} = \{0,1,4\}$, $\Longleftrightarrow$ Dynamics between the sub-systems
$SUB(0)$
$SUB(1)$
$SUB(4)$
a b c d e f g h i
$\frac{i}{h+i}$
$\frac{h}{h+i}$
1

Figure III.2: Decomposition of an example CTMC Z into subsystems and the embedded DTMC. The labels $a, b, \ldots, i$ denote transition rates. Since the two graphs on the bottom represent DTMCs, their transition labels denote one-step transition probabilities.

III.1.2 Decomposition (mathematical background)

Define the stopping time T_n as the time instant at which Z enters $E^{(X)}$ for the n-th time. Then $X = (X_n) = (Z_{T_n})$ is a DTMC embedded in Z. Suppose that X is irreducible and aperiodic recurrent. Furthermore, let $\pi = (\pi(0), \pi(1), \ldots)$ be an invariant measure for X.

The kernel of Z is defined as the set of conditional probabilities

$$K_t(x, A) = \mathbb{P}(Z_t \in A, T_1 > t | Z_0 = x),$$

for $x \in E^{(x)}$, $t \geq 0$ and $A \subseteq E$.

We write

$$\mathbb{E}_x[T_1] = \int_0^\infty K_t(x, E)dt \qquad \text{and} \qquad \mathbb{E}_{A|x} = \int_0^\infty K_t(x, A)dt.$$

The following theorem is taken from [14].

Theorem 1. *Provided that the function $t \rightarrow K_t(x, A)$ is Riemann integrable for each*

$x \in E^{(X)}$, the following assertion holds:

$$\lim_{t\to\infty} \mathbb{P}(Z_t \in A | Z_0 = x) = \frac{\sum_{x\in E^{(X)}} \pi(x)\mathbb{E}_{A|x}}{\sum_{x\in E^{(X)}} \pi(x)\mathbb{E}_x[T_1]}. \tag{III.2}$$

From this, equation (III.1) is verified by the following:

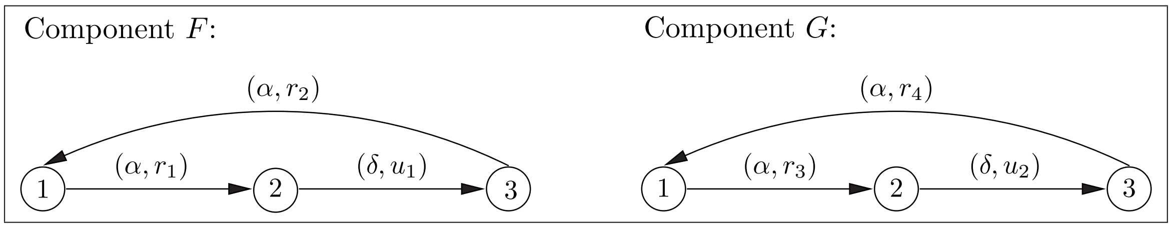
- $\mathbb{E}_x[T_1]$ is the expected time instant at which the next embedded state is reached, provided Z was initially in state x, i.e., in the above terminology this is the mean sojourn time $T(x)$ of $SUB(x)$.
- $\mathbb{E}_{A|x}$ is the total expected time that Z spends in A before reaching the next embedded state, provided Z was initially in state x, i.e., in the above terminology this is the expected total time $T(x)(A)$ which $SUB(x)$ spends in A.
- Obviously, the steady-state probability $\pi(x)$ coincides with the relative visiting frequency of $SUB(x)$.

III.1.3 An Illustrating Example.

Consider the PEPA expression $C = F \underset{L}{\bowtie} G$, with $L = \{\alpha\}$, and

$$\begin{aligned} F_1 &= (\alpha, r_1).F_2 & G_1 &= (\alpha, r_3).G_2 \\ F_2 &= (\delta, u_1).F_3 & G_2 &= (\delta, u_2).G_3 \\ F_3 &= (\alpha, r_2).F_1 & G_3 &= (\alpha, r_4).G_1 \end{aligned}$$

Fig. III.3 shows the labelled transition systems of the components F and G. The component F_i corresponds to the state i of the underlying CTMC, and analogously the state G_i corresponds to the state i of the underlying CTMC. In Fig. III.4 the CTMC underlying the composite component $C = F \underset{L}{\bowtie} G$ is shown, where in a composite state (i, j) the first element i refers to the CTMC underlying F and the element j refers to the CTMC underlying G.



$$\begin{tikzcd} 1 \arrow[r, "(\alpha, r_1)"] & 2 \arrow[r, "(\delta, u_1)"] & 3 \arrow[ll, bend right, "(\alpha, r_2)"'] \end{tikzcd} \qquad \begin{tikzcd} 1 \arrow[r, "(\alpha, r_3)"] & 2 \arrow[r, "(\delta, u_2)"] & 3 \arrow[ll, bend right, "(\alpha, r_4)"'] \end{tikzcd}$$

Figure III.3: Example processes F and G.

Suppose we want to calculate the steady-state probability that C is in the state $(2, 3)$, i.e., $A = \{(2, 3)\}$. The first step in applying the general decomposition scheme is the

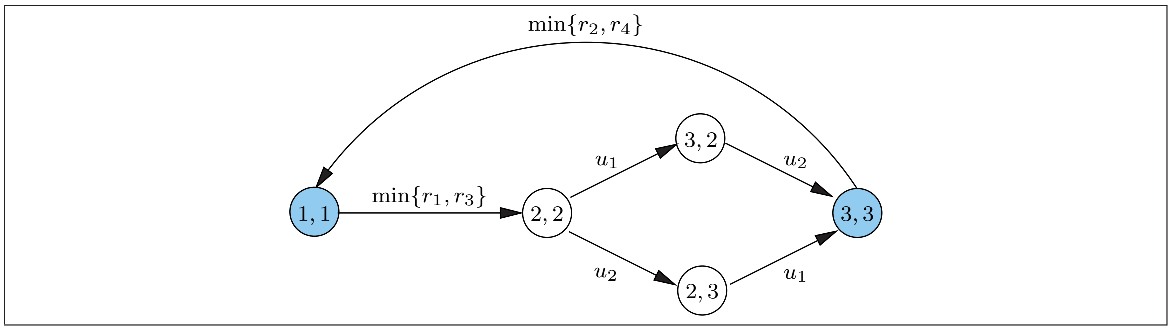


Figure III.4: CTMC underlying the composite PEPA component $C = F \underset{\{\alpha\}}{\bowtie} G$.

choice of the set of embedded states $E^{(X)}$. Here, in an ad-hoc[1] way we set $E^{(X)} = \{(1,1),(2,2),(3,3)\}$.

The embedded DTMC The set of embedded states $E^{(X)}$ defines an embedded DTMC. It can immediately be seen that the transition probabilities of this embedded DTMC are given by $\mathbb{P}((1,1) \rightarrow (2,2)) = 1$, $\mathbb{P}((2,2) \rightarrow (3,3)) = 1$ and $\mathbb{P}((3,3) \rightarrow (1,1)) = 1$. The steady-state probabilities of the DTMC are then $\pi_{(1,1)} = \pi_{(2,2)} = \pi_{(3,3)} = 1/3$.

The subsystems Every state $\mathbf{x} \in E^{(X)}$ defines a subsystem $SUB(\mathbf{x})$. The subsystems of the composite component are indicated in Fig. III.5.

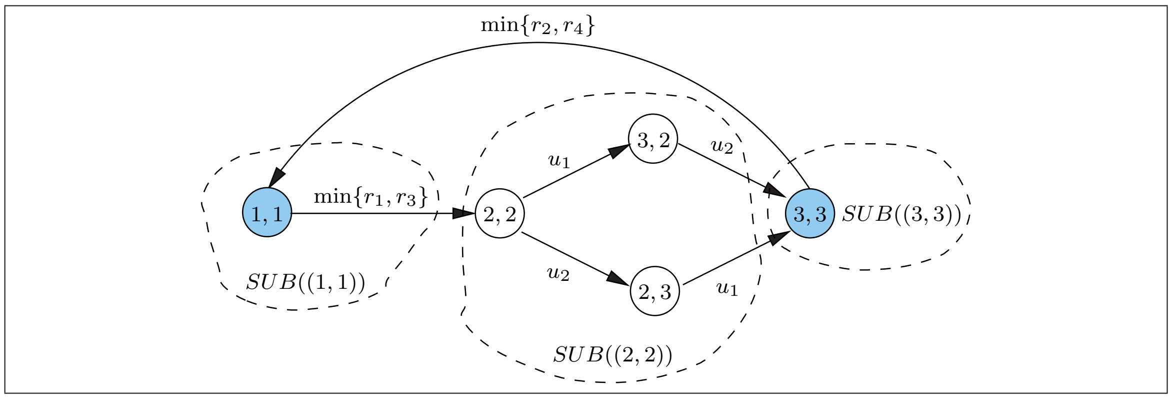


Figure III.5: The subsystems of the CTMC underlying the composite PEPA component.

For every subsystem $SUB(\mathbf{x})$, $\mathbf{x} \in E^{(X)}$, we have to calculate the mean sojourn time $T(\mathbf{x})$ and the expected total time $T(\mathbf{x})(A)$ which $SUB(\mathbf{x})$ spends in $A = \{(2,3)\}$. For our

[1]In order to apply the general decomposition scheme and for the purpose of illustrating it, the choice of $E^{(X)}$ can be arbitrary. In anticipation of the two-level decomposition scheme which is to be discussed in detail from section III.2 onwards, the set $E^{(X)}$ contains just the global synchronising states and the states which can be reached immediately after a synchronisation (cp. Fig III.4).

simple example, this can be easily done by just looking at the subsystems in Fig. III.5:

$$SUB((1,1)): \qquad T((1,1)) = \frac{1}{\min\{r_1, r_3\}}, \qquad T((1,1))(A) = 0.$$

$$\begin{aligned} SUB((2,2)): \qquad T((2,2)) &= \frac{1}{u_1+u_2} + \frac{u_1}{u_1+u_2}\frac{1}{u_2} + \frac{u_2}{u_1+u_2}\frac{1}{u_1}, \\ T((2,2))(A) &= \frac{u_2}{u_1+u_2}\frac{1}{u_1}. \end{aligned}$$

$$SUB((3,3)): \qquad T((3,3)) = \frac{1}{\min\{r_2, r_4\}}, \qquad T((3,3))(A) = 0.$$

Substituting these values and the steady-state probabilities $\pi(\mathbf{x}) = \frac{1}{3}$, $\mathbf{x} \in E^{(X)}$, in formula (III.1) yields

$$\mathbb{P}(C \in A) = \frac{\frac{u_2}{u_1+u_2}\frac{1}{u_1}}{\frac{1}{\min\{r_1,r_3\}} + \left(\frac{1}{u_1+u_2} + \frac{u_1}{u_1+u_2}\frac{1}{u_2} + \frac{u_2}{u_1+u_2}\frac{1}{u_1}\right) + \frac{1}{\min\{r_2,r_4\}}}.$$

If, e.g., we take the concrete values $r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, u_1 = 1, u_2 = 2$, we obtain the steady-state probability

$$\mathbb{P}(C \in A) = \frac{1}{4}.$$

This example only serves the purpose of somewhat illustrating the basic proceeding. Note that here we explicitly constructed the CTMC underlying the composite PEPA component C and afterwards extracted the embedded DTMC from that CTMC, as well as the subsystems. The aim of the following sections is to get access to the embedded DTMC and the quantities of the related subsystems in a compositional fashion, i.e., to extract them from the PEPA descriptions of the local components, hence, without first generating the overall CTMC.

III.2 The Algorithm (Parts 1, 2 and 3)

In this section we give a general outline of an algorithm which computes steady-state probabilities of a composite PEPA component C. We anticipate that this approach consists of the exploitation of two levels of compositionality, hence, we call it the *two-level decomposition scheme* (2-LDS).

If C possesses an underlying ergodic CTMC, say Z, then Z can be solved for steady-state probabilities. Hence, for any subset A of the state space, the steady-state probability

$\mathbb{P}(Z \in A)$ can be calculated by applying the decomposition scheme of the previous section to Z. We will refer to this as the first level of compositionality.

First Level of Compositionality The first level of compositionality obviously results from the decomposition of the CTMC Z into several subsystems. The solutions for these subsystems are obtained in isolation. Together with the solution of the DTMC embedded in Z these solutions are combined via formula (III.1) to yield the desired steady-state probability. The choice of the set $E^{(X)}$ of embedded states determines the subsystems $SUB(x)$, $x \in E^{(X)}$, as well as the embedded DTMC X.

The second level of compositionality deals with the structure of the subsystems. We briefly discuss the basic idea – a thorough treatment is given later.

Second Level of Compositionality The concurrency inherent to a composite PEPA component C is reflected in the underlying CTMC Z in one way or the other, and consequently also in the subsystems of Z. As we will come to know, specific conditions imposed on the structure of the composite component C, as well as a proper choice of the set of embedded states $E^{(X)}$ can be exploited to represent the subsystems of Z as a purely parallel (i.e., no interaction) composition of several CTMCs. This parallelism will then yield a compositional solution method for the values $T(x)$ and $T(x)(A)$ of the subsystem $SUB(x)$, for all $x \in E^{(X)}$. This makes the approach just sketched less vulnerable to the problem of state space explosion.

Below, a roadmap of the two-level decomposition scheme is given. It is subdivided into three parts 1, 2 and 3 which are discussed in detail in subsequent sections.

The Two-Level Decomposition Scheme: Outline

(Part 1) Make sure that the composite PEPA component C meets certain general requirements which ensure the existence of a steady-state solution and the exploitability of the two levels of compositionality.

(Part 2) Identify a suitable subset $E^{(X)}$ of the state space of C. Given the set $E^{(X)}$, construct the subsystems $SUB(x)$, $x \in E^{(X)}$, as well as the corresponding embedded DTMC X. Calculate the steady-state distribution π of the embedded DTMC X.

(Part 3) For every subsystem $SUB(x)$, $x \in E^{(X)}$, calculate the mean sojourn time $T(x)$. In addition, calculate the expected total time spent in the set A, i.e., $T(x)(A)$.

Then, evaluate:

$$\mathbb{P}(Z \in A) = \frac{\sum_{x \in E^{(X)}} \pi(x) T(x)(A)}{\sum_{x \in E^{(X)}} \pi(x) T(x)}. \tag{III.3}$$

III.3 The Bohnenkamp Method

In his dissertation [5], Bohnenkamp proposed a solution method for MPAs which shares many similarities to our approach but also exhibits a few differences. He invented the MPA $\mathscr{YAWN}$ in order to transport his ideas. The similarities between his method and our approach are strong enough to identify our approach as a direct extension of his work. Before delving into the details of our algorithm in the subsequent sections, we briefly discuss the similarities and differences between Bohnenkamp's approach and our algorithm.

Bohnenkamp's method solves a semi-Markov chain (SMC) which is embedded in the original CTMC. To achieve this he decomposes the CTMC which underlies a given MPA model into several subsystems $SUB(x)$, $E^{(X)}$, in the same way as our algorithm. The set $E^{(X)}$ is the set of embedded states which are associated with global synchronisation points in the MPA model. As explained in section III.1, the set of embedded states $E^{(X)}$ defines an embedded DTMC which, under suitable assumptions, possesses a steady-state distribution π. This embedded DTMC, in turn, defines a semi-Markov chain (SMC) with state space $E^{(X)}$ which is embedded in the original CTMC. $T(x)$, $x \in E^{(X)}$, was introduced as the mean sojourn time of the subsystem $SUB(x)$. At the same time, it is the mean sojourn time in the state x of the embedded SMC. Then, the steady-state probabilities of the embedded SMC are given by (see [14]):

$$smc(x) := \frac{\pi(x)T(x)}{\sum_{k\in E^{(X)}} \pi(k)T(k)}, \qquad x \in E^{(X)}.$$

Bohnenkamps method consists of two parts:

- Efficient computation of the quantities $T(x)$, $x \in E^{(X)}$, by exploitation of the second level of compositionality. This is discussed in more detail in section IV.2.4.
- Computation of the steady-state distribution π of the embedded DTMC.

An MPA model which is the cooperation of m components possesses an m-dimensional underlying CTMC. Of course, also the embedded DTMC is m-dimensional. In the MPA $\mathscr{YAWN}$ used by Bohnenkamp all probabilistic choices which occur in a sequential component are independent of probabilistic choices in other sequential components. This independence allowed to establish m (local) embedded DTMCs for the sequential components, and to represent the steady-state distribution of the global embedded DTMC as the tensor product of the local steady-state distributions. In our approach, we will loosen the restrictions for the considered MPA (or MPA model), in the sense that probabilistic choices within different sequential components may be dependent on each other. As a result we are no longer able to give a product-form solution for the embedded DTMC.

III.4 The Algorithm: Part 1 – The Requirements

In order to derive steady-state probabilities from a PEPA component C in the way described above, we ask the component to possess the following properties:

Condition 1: The component is cyclic. This condition ensures that the CTMC underlying the PEPA component C is ergodic, hence, it possesses a unique stationary distribution.

Condition 2: All local components, say $C_1, \ldots, C_m$, synchronise over the same set L. If C is composed of m concurrent components $C_1, \ldots, C_m$, this condition implies that C can be written as

$$C = C_1 \underset{L}{\bowtie} C_2 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} C_m.$$

Condition 3: There exists no choice between synchronising and non-synchronising activities. This means language terms of the form of $(\alpha, r_1).C_1 + (\beta, r_2).C_2$, with $\alpha \in L$ and $\beta \notin L$, are prohibited. This condition eliminates the possibility that a global state x can decide whether to be a synchronising or a non-synchronising state. Furthermore, it ensures that a local component always must wait for synchronisation, once it has reached a local synchronising state.

The consequence of the conditions 2 and 3 is that every local component is involved in every synchronisation, i.e., the local components can only perform barrier synchronisations. If the local components are currently not involved in a synchronisation they evolve independently of each other. As soon as a local component reaches a synchronising state, i.e., a synchronising activity is enabled in that component, it is ready to synchronise. But in order for the synchronising activity to be executed this component has to wait until all other local components are ready to synchronise on their part. If, finally, all local components are ready to synchronise, the synchronisation can take place. The time duration of the synchronising (or shared) activity is the same for all local components. After execution of the synchronising activity, the local components again start to evolve independently of each other. A sample path of such a system is shown in Fig. III.6.

Now, let S be the set of synchronising states and let NS be the set of states which can be reached immediately after a synchronising state and are not synchronising states themselves. If the global system is decomposed into subsystems along the set $S \cup NS$ – i.e., every $SUB(x)$, with $x \in S \cup NS$, is a subsystem – we recognise the following important

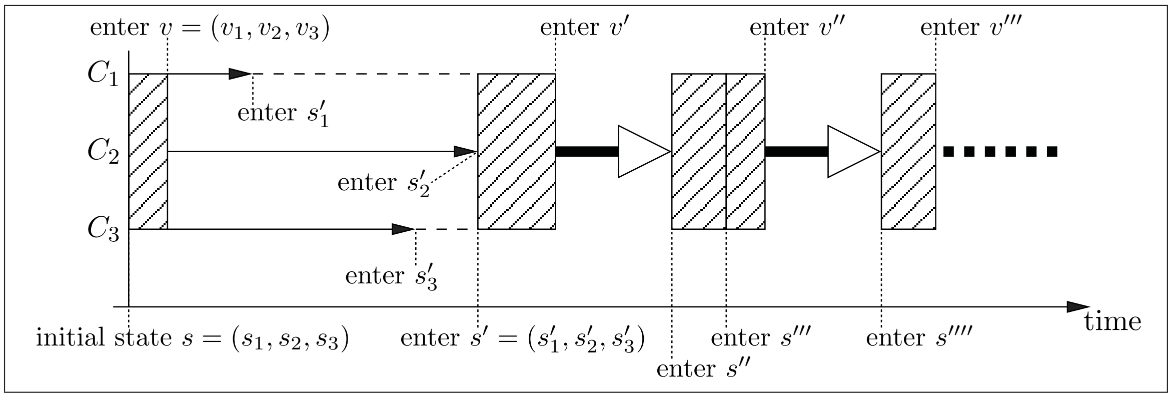


Figure III.6: MPA model with barrier synchronisations. The thick arrows are an aggregated representation of the independent evolution of the marginal processes Y_1, Y_2 and Y_3. $S = \{s, s', s'', \ldots\}$. $NS = \{v, v', v'', \ldots\}$.

properties: If $x = (x_1, \ldots, x_m) \in S$, then $SUB(x)$ consists only of the synchronising state x and the outgoing synchronising transitions. If $x \in NS$, then $SUB(x)$ is characterised by the independent evolution of the m marginal processes. In Fig. III.6 the subsystem $SUB(v)$ would describe the independent and parallel evolution of the three components C_1, C_2 and C_3 from the time instant of entering state $v = (v_1, v_2, v_3)$ until the time instant of entering the embedded successor state $s' = (s'_1, s'_2, s'_3) \in S$. This independence inside of given subsystems will be crucial in efficiently solving these subsystems, and as a consequence in solving the entire system.

2-LDS: The Set $E^{(X)}$ of Embedded States

Let S be the set of synchronising states and let NS be the set of states which can be reached immediately after a synchronising state and are not synchronising states themselves. Then, the set of embedded states is chosen as

$$E^{(X)} = S \cup NS.$$

III.5 The Algorithm: Part 2 – The Embedded DTMC

The DTMC X which is embedded in the composite PEPA component C is an m-dimensional DTMC, hence, it can be written as $X = (X_1, \ldots, X_m)$. X_i is the projection of X onto its $i-$th component. The solution of $X = (X_1, \ldots, X_m)$ is given by the probability vector π, which (uniquely) satisfies $\pi P = \pi$. The topic of this section is the construction of X from the local PEPA components.

III.5.1 Establishing the Embedded DTMC

The embedded DTMC X is given by the state space $E^{(X)}$, the transition probabilities $\mathbb{P}(X(n+1) = \mathbf{x}'|X(n) = \mathbf{x})$, for $\mathbf{x}, \mathbf{x}' \in E^{(X)}$, as well as an initial state s. The state space $E^{(X)} = S \cup NS$ can be determined by a simple reachability analysis. That means, we start with the initial state s and determine all of its embedded successor states. Afterwards, their embedded successor states are determined, and so on. During this reachability analysis also the transition probabilities between embedded states are determined.

An embedded state is either in S (synchronising state) or in NS (non-synchronising state). We anticipate the following, for an embedded state $\mathbf{x}$: If $\mathbf{x} \in S$, the embedded successor states of $\mathbf{x}$ and the transition probabilities from $\mathbf{x}$ to the embedded successor states can be obtained by simply applying the PEPA cooperation operator to the state $\mathbf{x}$. If $\mathbf{x} \in NS$, the transition probabilities from $\mathbf{x}$ to the embedded successor states can be obtained from the local components. Hence, in an algorithm which performs a reachability analysis on $E^{(X)}$ the distinction whether $\mathbf{x} \in S$ or $\mathbf{x} \in NS$ is necessary.

The following subprocedure 1 treats the case, where $\mathbf{x}$ is a synchronising state, and subprocedure 2 treats the case, where $\mathbf{x}$ is an embedded non-synchronising state.

Subprocedure 1: $\mathbf{x} \in S$

Suppose $\mathbf{x} = (x_1, \ldots, x_m) \in S$, i.e., $\mathbf{x}$ is a synchronising state. In order to obtain the possible successor states of $\mathbf{x}$, as well as the corresponding transition rates, one has to do the following:

2-LDS: The Embedded DTMC – Subprocedure 1

For a given $\mathbf{x} = (x_1, \ldots, x_m) \in S$:

- For every $\alpha \in L$ apply the PEPA cooperation-operator to $x_1, \ldots, x_m$. This yields the set of successor states V.
- For every $\mathbf{x}' = (x'_1, \ldots, x'_m) \in V$ determine the rate R of the transition $\mathbf{x} \to \mathbf{x}'$. Assume the transition is labelled with α.
  (a) Determine the rate r_i of the marginal transition $x_i \to x'_i$ equipped with label α, as well as the apparent rate $r_\alpha(x_i)$.
  (b) The transition rate of $\mathbf{x} \to \mathbf{x}'$ is given by $R = \min_i\{r_\alpha(x_i)\} \prod_i \frac{r_i}{r_\alpha(x_i)}$.

Explanation of (a) and (b):

(a) $r_\alpha(x_i)$ is the total rate of all α-transitions out of the marginal state x_i and r_i is the rate of all α-transitions from x_i to the particular marginal successor state x'_i. The fraction $\frac{r_i}{r_\alpha(x_i)}$ is then the probability that of all possible marginal successor states reachable by α-transitions x'_i is actually chosen.

(b) The total rate of α-transitions out of the synchronising state $\mathbf{x}$ is given by the minimum of all outgoing rates of α-transitions, i.e., $\min_i\{r_\alpha(x_i)\}$. $\frac{r_i}{r_\alpha(x_i)}$ is the probability that (of all possible α-transitions) the marginal transition $x_i \to x'_i$ takes place, hence, $\prod_i \frac{r_i}{r_\alpha(x_i)}$ is the probability that (of all possible α-transitions) $\mathbf{x}'$ will be the successor state of $\mathbf{x}$. Thus, R is indeed the transition rate of $\mathbf{x} \to \mathbf{x}'$.

Subprocedure 2: $\mathbf{x} \in NS$

Now suppose $\mathbf{x} = (x_1, \ldots, x_m) \in NS$, i.e., $\mathbf{x}$ is an embedded state which (a) is a state to be reached immediately after leaving a synchronising state, and (b) is not a synchronising state itself. As above, the aim here is to determine the embedded successor states of $\mathbf{x}$ and the transition probabilities to the successor states.

The state $\mathbf{x}$ is the first state after a synchronising transition. That means, as soon as the system enters the state $\mathbf{x}$, the m processes start to evolve independently of each other until they reach the next embedded state which must be a synchronising state, say $\mathbf{x}' = (x'_1, \ldots, x'_m)$. Assume that for the i-th component the probability that x'_i is the next embedded state to be reached is given by $\mathbb{P}(x_i \to x'_i)$. Since the marginal successor states are reached independently by the m components, we can immediately state that the probability $\mathbb{P}(\mathbf{x} \to \mathbf{x}')$ is the product of marginal probabilities:

$$\mathbb{P}(\mathbf{x} \to \mathbf{x}') = \prod_i \mathbb{P}(x_i \to x'_i). \tag{III.4}$$

Furthermore, if V_i is the set of reachable embedded states of the i-th component if started in x_i, then $\times_i V_i$ is the set of reachable embedded states of the entire process if started in $\mathbf{x} = (x_1, \ldots, x_m)$.

Calculating the marginal sets of reachable embedded states as well as the transition probabilities is fairly easy: Consider the state $\mathbf{x} = (x_1, \ldots, x_m) \in NS$. Since any next embedded state $\mathbf{x}' = (x'_1, \ldots, x'_m)$ must be a synchronising state, the candidates for the next local embedded states in component C_i are the local synchronising states $\in S_i$. For each $x'_i \in S_i$, the local transition probability $\mathbb{P}(x_i \to x'_i)$ is the probability that in component C_i the state x'_i is reached before any other state of S_i is reached, provided C_i was initially in x_i. Such a probability is sometimes referred to as ruin probability. It is computed by at first declaring the states of S_i as absorbing states. Now, the probability to become absorbed in the state $x'_i \in S_i$ is the wanted transition probability from x_i to x'_i.

2-LDS: The Embedded DTMC – Subprocedure 2

Let $\mathbf{x} = (x_1, \dots, x_m) \in NS$ be a given global embedded state. For $i = 1, \dots, m$, Let $\hat{Q}_i$ be the absorbing generator matrix which results from declaring all synchronising states in component C_i as absorbing states. Let $init_i$ be a probability distribution, where all probability mass is gathered in state x_i, i.e., $init_i(x_i) = 1$ and $init_i(k) = 0$, for $k \neq x_i$. Then compute the vector of absorption probabilities:

$$p_i = \lim_{t \to \infty} init_i \cdot e^{t\hat{Q}_i}$$

Now, for all $x'_i \in S_i$, $p_i(x'_i)$ is the transition probability from x_i to the next local synchronising state x'_i. Define V_i by

$$x'_i \in V_i \Longleftrightarrow p_i(x'_i) > 0.$$

Then the set of embedded successor states of $\mathbf{x}$ is given by

$$V := \times_{i=1}^{m} V_i.$$

Furthermore

$$\mathbb{P}(\mathbf{x} \to \mathbf{x}') = \prod_{i=1}^{m} p_i(x'_i), \text{ for all } \mathbf{x}' = (x'_1, \dots, x'_m) \in V.$$

Remark 2. One might wonder whether the explicit determination of the transition probabilities, or equivalently the explicit construction of the transition probability matrix P, is contrary to our initial wish to circumvent the state space explosion problem. To answer this question, notice that the dimension of P equals the number of embedded states $E^{(X)} \subseteq S \cup NS$. This number typically is small, even if the state space E of the CTMC Y (underlying the composite PEPA component C) is huge.

III.5.2 The Detailed Algorithm

Below is the detailed algorithm to determine the state space and the transition probabilities of the embedded DTMC. Basically this algorithm is a depth-first search which explores all reachable embedded states, i.e., the states which finally constitute the set $E^{(X)}$. In order to determine the embedded successor states of the actual state $\mathbf{x}$, as well as the transition probabilities to the successor states, it is distinguished whether $\mathbf{x} \in S$ or $\mathbf{x} \in NS$.

The set $EXPLORED$ contains all the states which have already been explored by the reachability analysis. Initially $EXPLORED = \emptyset$. In the parameter list, the matrix $\hat{Q}_i$

is the (local) absorbing generator which results from declaring all synchronising states in the local component C_i as absorbing states.

```
Procedure TransitionProbabilities(Matrix $\hat{Q}_1$, ..., Matrix $\hat{Q}_m$,state x)
1  if $\mathbf{x} \in S$ then                                      /* Subprocedure 1 */
2      Calculate the derivations of $\mathbf{x} = x_1 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} x_m$: This yields the set $V$ of
       successor states (derivations) of $\mathbf{x}$ and the transition probabilities to the
       successor states;
3  else                                                           /* Subprocedure 2 */
4      for $i = 1 \ldots m$ do
5          let $init_i$ be a probability vector, with all mass gathered in the state $x_i$;
6          compute $p_i = \lim_{t\to\infty} init_i \cdot e^{t\hat{Q}_i}$;
7      Determine set $V$ of successor states of $\mathbf{x}$:
       $\mathbf{x}' = (x'_1, \ldots, x'_m) \in V \Longleftrightarrow \prod_i p_i(x'_i) > 0$;
8      (The transition probabilities are implicitly calculated above);
9  forall $\mathbf{x}' \in V$ do
10     if $\mathbf{x}' \notin EXPLORED$ then
11         $EXPLORED := EXPLORED \cup \{\mathbf{x}'\}$;
12         call TransitionProbabilities($\hat{Q}_1$, ..., $\hat{Q}_m$, $\mathbf{x}'$);
```

III.6 The Algorithm: Part 3 – The Subsystems

On the one hand, the set of embedded states $E^{(X)}$ defines an embedded DTMC, which has been the topic of the preceding section. On the other hand, every state $\mathbf{x} \in E^{(X)}$ defines a subsystem $SUB(\mathbf{x})$. The solution of such a subsystem $SUB(\mathbf{x})$, i.e., the computation of the sojourn time $T(\mathbf{x})$ and the expected total time $T(\mathbf{x})(A)$ which the subsystem spends in set A, is the matter of the current section.

III.6.1 Solving the Subsystems

The solution method for the subsystem $SUB(\mathbf{x})$ will depend on whether $\mathbf{x} \in S$ or $\mathbf{x} \in NS$. Hence, in analogy to the previous section we introduce two subprocedures. The subprocedure 1 treats the case, where $\mathbf{x} \in S$, and the subprocedure 2 treats the case, where $\mathbf{x} \in NS$.

Subprocedure 1: $\mathbf{x} \in S$

If $\mathbf{x} \in S$, then the subsystem $SUB(\mathbf{x})$ consists only of the state $\mathbf{x}$ itself and its outgoing synchronising transitions. This is the case because by definition the successor state of $\mathbf{x}$ is a state to be reached immediately after a synchronising transition, i.e., the successor state is an embedded state and therefore is the entrance state of the subsequent subsystem. Thus, the rate $\lambda(\mathbf{x})$ out of state $\mathbf{x}$ is the rate out of the subsystem $SUB(\mathbf{x})$. It is given by the sum of the rates of all outgoing synchronising transitions which were already obtained as a by-product of Subprocedure 1 in section III.5.1. Hence, the mean sojourn time in $SUB(\mathbf{x})$ is given by $1/\lambda(\mathbf{x})$.

2-LDS: The Subsystems – Subprocedure 1

Let $\lambda(\mathbf{x})$ be the total rate out of the synchronising state $\mathbf{x} \in S$. Then

$$T(\mathbf{x}) = \frac{1}{\lambda(\mathbf{x})}.$$

Furthermore, it can immediately be stated that

$$T(\mathbf{x})(A) = \begin{cases} 0 & \text{if } \mathbf{x} \notin A \\ \frac{1}{\lambda(\mathbf{x})} & \text{if } \mathbf{x} \in A \end{cases}.$$

Subprocedure 2: $\mathbf{x} \in NS$

Here, we examine a subsystem $SUB(\mathbf{x})$, where $x \in NS$, i.e., the entrance state $\mathbf{x}$ is a non-synchronising state. The subsystem $SUB(\mathbf{x})$ describes the behaviour of the composite component C, with the initial state set to $\mathbf{x}$, until the next embedded state ($\in E^{(X)}$) is reached.

We again take a look at what happens in C if started in the state $\mathbf{x}$. In $\mathbf{x} = (x_1, \ldots, x_m)$ the m local components start to evolve independently of each other. If a local component reaches a local synchronising state, it must stay there until all other components have reached their local synchronising states. The time instant where the slowest component has reached its local synchronising state is the time instant where the synchronisation begins; it is the time instant of entering the next embedded state, i.e., the time instant of leaving the subsystem $SUB(\mathbf{x})$.

Now, consider the following modified system: Declare local synchronising states as absorbing states. Then the global component would become absorbed if all local components have become absorbed. The time instant of absorption of the modified system would then correspond to the time instant of entering the next embedded state of the original system. That means until that time instant the two systems behave identically. After that time instant, of course, the original system moves ahead whereas the modified system stays in the absorbing state forever. This situation is illustrated in Fig. III.7.

The following two points are obvious:

- The quantity $T(\mathbf{x})$ (sojourn time in $SUB(\mathbf{x})$) is then the expected absorption time of the modified absorbing system. The quantity $T(\mathbf{x})(A)$ (expected total time which $SUB(\mathbf{x})$ spends in A) is the expected total time which the modified absorbing system spends in A before absorption.
- In the original system, the local components behave independently until reaching the next synchronising state. Hence, also the modified absorbing components are independent of each other until reaching the absorbing state. Once a modified local component has reached a local absorbing state, it stays there forever, i.e., independently of the other components. As a conclusion, the modified absorbing local components are independent of each other.

2-**LDS: The Subsystems – Subprocedure 2**

Let Y_i be the absorbing Markov chain underlying the modified (absorbing) component C_i, and let $Y := (Y_1, \dots, Y_m)$ be the absorbing joint CTMC, with initial state $\mathbf{x} = (x_1, \dots, x_m) \in NS$.

- $T(\mathbf{x})$ is the mean time to absorption of the absorbing joint CTMC $Y := (Y_1, \dots, Y_m)$.
- $T(\mathbf{x})(A)$ is the expected total time Y spends in A **before absorption**. That means:
  - if A contains only transient states (i.e., no synchronising states): $T(\mathbf{x})(A)$ is the expected total time Y spends in A. Note, that $A = \times_{i=1}^{m}$ is transient iff at least one of the A_i contains only transient states.
  - if A contains only absorbing (synchronising) states: $T(\mathbf{x})(A) = 0$.

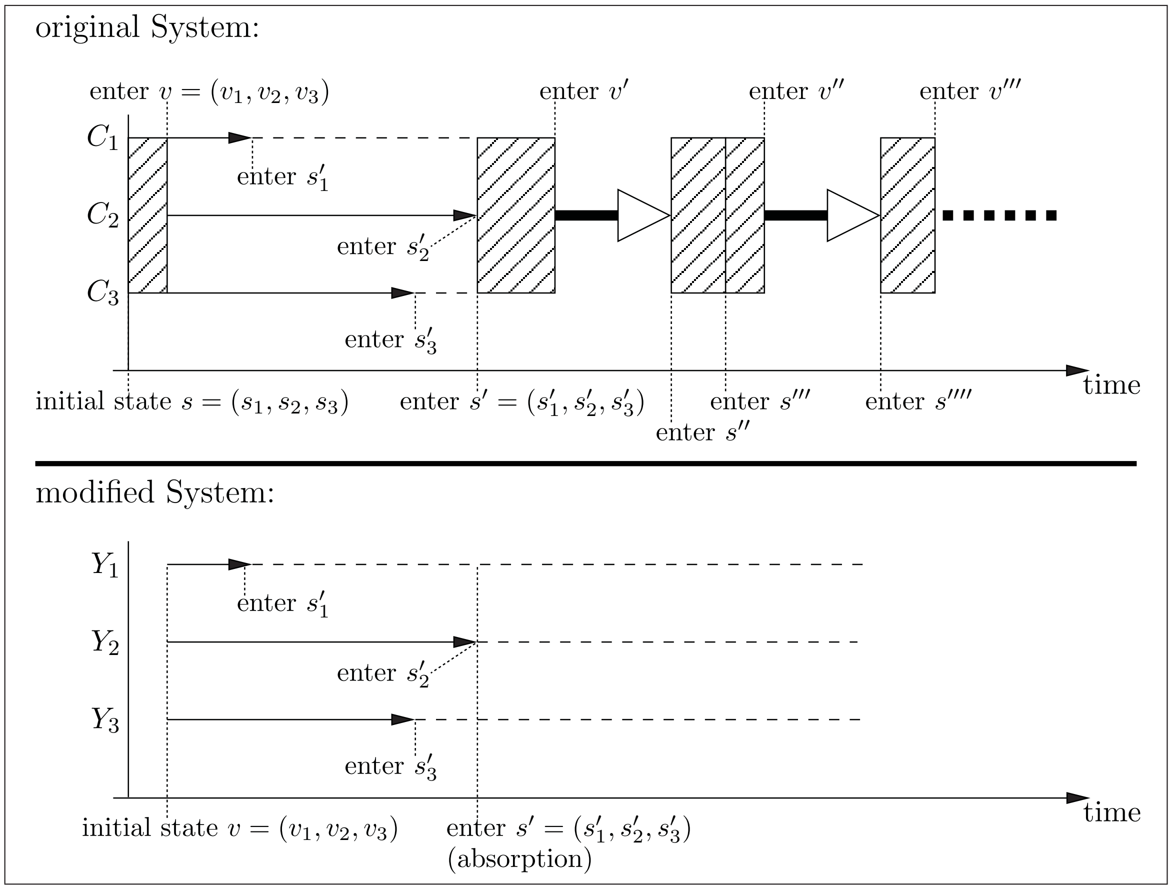


Figure III.7: Upper picture: MPA model with barrier synchronisations (reproduction of Fig. III.6). Lower picture: In the MPA model the synchronising states are declared as absorbing states. If in the modified model we chose the initial state v, the behaviour until reaching the absorbing state is probabilistically identical to the behaviour of the original (upper) model until reaching the synchronising state.

III.6.2 The Detailed Algorithm

Below is the detailed algorithm to determine the values $T(\mathbf{x})$ and $T(\mathbf{x})(A)$, for all $\mathbf{x} \in E^{(X)}$. In the parameter list, the matrix $\hat{Q}_i$ is the (local) absorbing generator which results from declaring all synchronising states in the local component C_i as absorbing states. Note, that we assume that either $A \subseteq S$ or $A \cap S = \emptyset$. That means, either A consists only of synchronising states, or doesn't contain any synchronising states. In lines 12 and 16 compositional uniformisation can be used to compute the values $T(\mathbf{x})$ and $T(\mathbf{x})(A)$, which is to be discussed in Chapter IV.

```
Procedure ExpectedTotalTimes(Matrix $\hat{Q}_1$, ..., Matrix $\hat{Q}_m$, set $A_1$, ..., set $A_m$)
1  $A := \times_{i=1}^{m} A_i$;
2  Let $Y_i$ be the CTMC defined by the absorbing generator $\hat{Q}_i$;
3  Let $Y := (Y_1, \dots, Y_m)$ be the absorbing joint CTMC;
4  forall $\mathbf{x} = (x_1, \dots, x_m) \in E^{(X)}$ do
5      if $\mathbf{x} \in S$ then
6          Compute $T(\mathbf{x})$ by deriving $x_1 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} x_m$;
7          if $\mathbf{x} \in A$ then
8              $T(\mathbf{x})(A) := T(\mathbf{x})$;
9          else
10             $T(\mathbf{x})(A) := 0$;
11     else                                                    /* $\mathbf{x} \in NS$ */
12         Compute $T(\mathbf{x})$;
           /* $T(\mathbf{x}) =$ mean time to absorption of $Y$, with initial state $\mathbf{x}$ */
13         if $A \subseteq S$ then
14             $T(\mathbf{x})(A) := 0$;
15         else                                                /* $A \cap S = \emptyset$ */
16             Compute $T(\mathbf{x})(A)$;
               /* $T(\mathbf{x})(A) =$expected total time in $A$ of $Y$, with initial
                  state $\mathbf{x}$.                                             */
```

III.7 A Generic Example

In order to illustrate our ideas of solving composite PEPA models, we investigate a simple example. Consider the composite PEPA component $C = F \underset{L}{\bowtie} G$, with $L = \{\alpha, \beta\}$, and

$$
\begin{aligned}
F_1 &= (\alpha, r_1).F_2 & & \\
F_2 &= (\alpha, r_2).F_3 + (\alpha, r_3).F_4 + (\beta, r_4).F_4 & G_1 &= (\alpha, r_5).G_2 \\
F_3 &= (\delta, u_1).F_5(\delta, u_3) & G_2 &= (\alpha, r_6).G_3 + (\beta, r_7).G_3 \\
F_5 &= (\delta, u_3).F_1 & G_3 &= (\alpha, u_5).G_4 \\
F_4 &= (\delta, u_2).F_6 & G_4 &= (\delta, u_7).G_1 + (\delta, u_6).G_2 \\
F_6 &= (\delta, u_4).F_1 & &
\end{aligned}
$$

The labelled transition systems of F and G are shown in Fig. III.8. In the system on the left hand side the state i corresponds to component F_i. On the right hand side state i corresponds to component G_i. Fig. III.9(a) shows the CTMC underlying the composite PEPA component $C = F \underset{L}{\bowtie} G$.

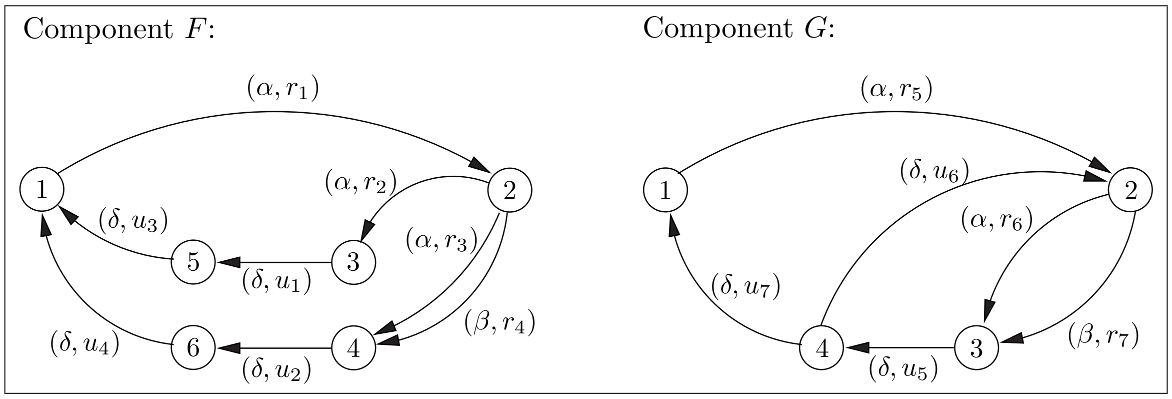


Figure III.8: Running example. The labelled transition systems of the components F and G.

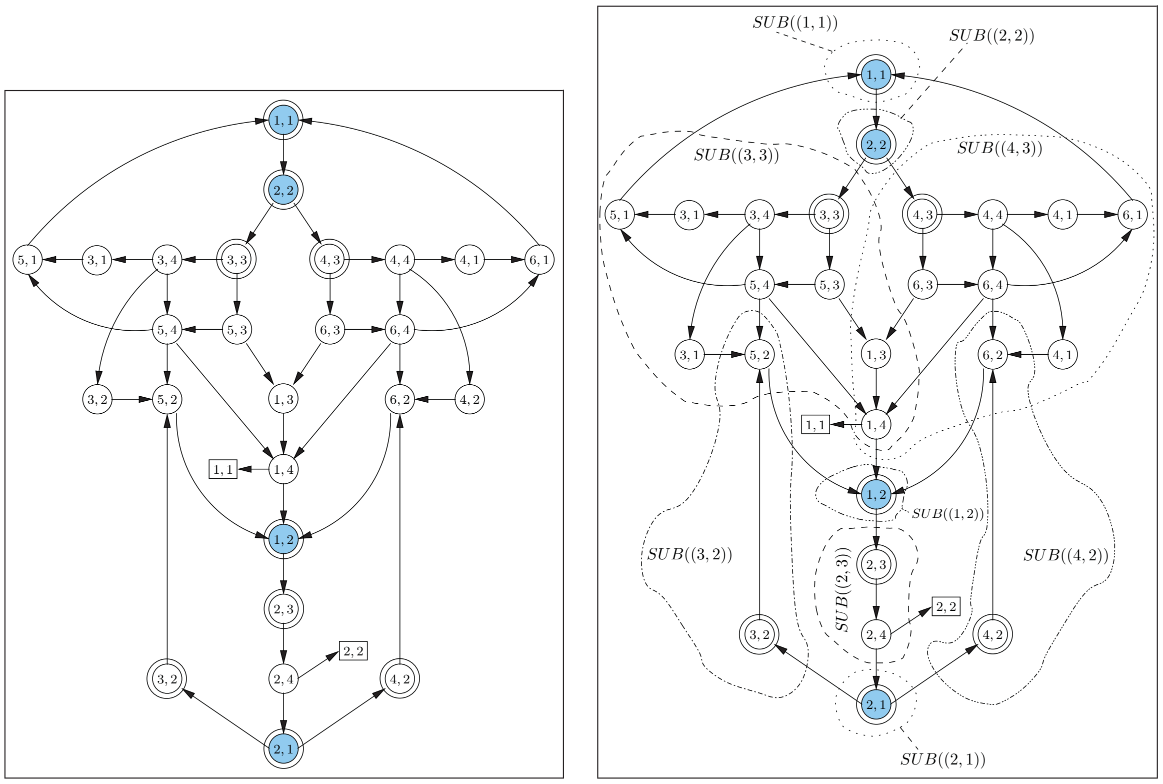
The generator matrices Q_F and Q_G of the CTMCs underlying the components F and G, as well as the generator matrices $\hat{Q}_F$ and $\hat{Q}_G$ which result from declaring the local synchronising states as absorbing are given by

$$
Q_F = \begin{pmatrix} -r_1 & r_1 & 0 & 0 & 0 & 0 \\ 0 & -(r_2+r_3+r_4) & r_2 & r_3+r_4 & 0 & 0 \\ 0 & 0 & -u_1 & 0 & u_1 & 0 \\ 0 & 0 & 0 & -u_2 & 0 & u_2 \\ u_3 & 0 & 0 & 0 & -u_3 & 0 \\ u_4 & 0 & 0 & 0 & 0 & -u_4 \end{pmatrix} \qquad \hat{Q}_F = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -u_1 & 0 & u_1 & 0 \\ 0 & 0 & 0 & -u_2 & 0 & u_2 \\ u_3 & 0 & 0 & 0 & -u_3 & 0 \\ u_4 & 0 & 0 & 0 & 0 & -u_4 \end{pmatrix},
$$

$$
Q_G = \begin{pmatrix} -r_5 & r_5 & 0 & 0 \\ 0 & -(r_6+r_7) & r_6+r_7 & 0 \\ 0 & 0 & -u_5 & u_5 \\ u_7 & u_6 & 0 & -(u_7+u_6) \end{pmatrix} \qquad \hat{Q}_G = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -u_5 & u_5 \\ u_7 & u_6 & 0 & -(u_7+u_6) \end{pmatrix}.
$$

The structure of the CTMC underlying the composite PEPA component C is indicated in Fig. III.9(a). States marked with double circles are embedded states, i.e., they constitute the set $E^{(X)} = S \cup NS$. Darked states are synchronising states. From Fig. III.9(a) it is seen that $S = \{(1,1),(1,2),(2,1),(2,2)\}$ and $NS = \{(2,3),(3,2),(3,3),(4,2),(4,3)\}$. Fig. III.9(b) illustrates the subsystems which correspond to the embedded states.

Of course, in the subsequent treatment of the example system the subsystems are established in a compositional fashion, i.e., without generating the CTMC underlying the composite PEPA component C.



(a) The CTMC underlying the composite PEPA component $C = F \underset{L}{\bowtie} G$.

(b) Decomposition of the overall CTMC into subsystems.

Figure III.9: *CTMC of the composite component C and subsystems. Double circles indicate embedded states, darked states are synchronising states.*

III.7.1 Example: The Embedded DTMC

The initial state of the composite component $C = F \underset{L}{\bowtie} G$ is the state $(1,1)$. It is easily seen that the set of synchronising states is given by $S = \{(1,1),(1,2),(2,1),(2,2)\}$. The set of states NS which can be reached is a bit more involved.

A call of the procedure TransitionProbabilities($\hat{Q}_F$, $\hat{Q}_G$, $(1,1)$) produces the reachability graph in Fig. III.10, where each node represents an embedded state (this reachability graph is easily verified by comparing it with the CTMC of the composite component C in Fig. III.9(a), where the states surrounded with a double circle are embedded states). If the edges of the reachability graph of embedded states are equipped with proper transition probabilities the embedded discrete time Markov chain of the component C is obtained. In order to illustrate how the transition probabilities are determined, and how the reachability graph is constructed, we take a closer look at the states $(1,1), (2,2) \in S$ and $(3,3), (2,3) \in NS$.

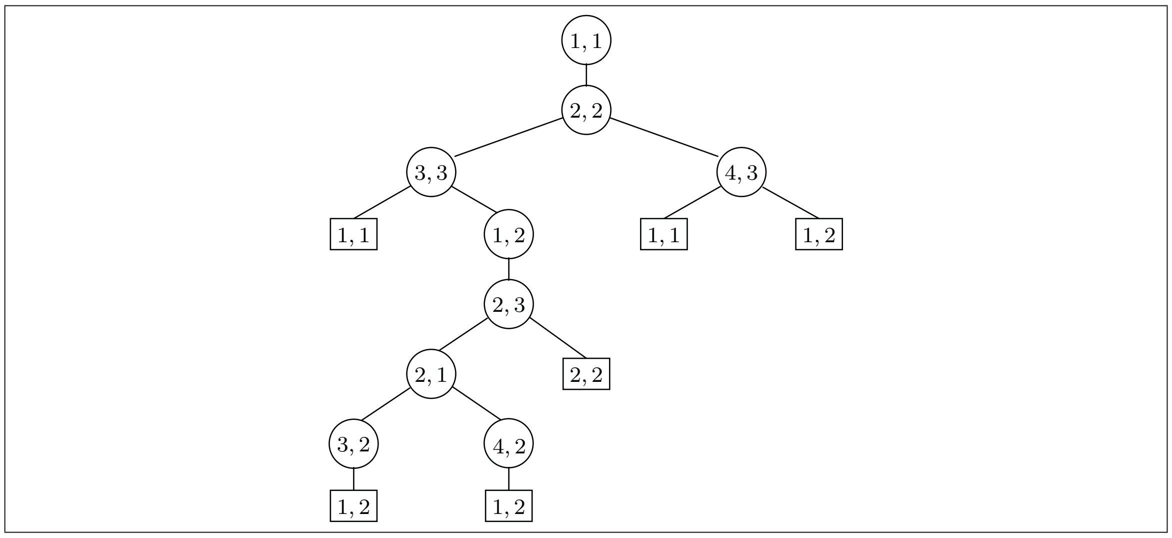


Figure III.10: Reachability graph generated by the procedure TransitionProbabilities$(1,1)$.

III.7.1.1 The synchronising states

state $(1,1) \in S$**:** The state $(1,1)$ corresponds to the PEPA component $F_1 \underset{L}{\bowtie} G_1$. Application of the cooperation operator to F_1 and G_1 (i.e., execution of the shared transition labelled with α) yields the component $F_2 \underset{L}{\bowtie} G_2$, or equivalently the state $(2,2)$. Since $(2,2)$ is the only possible successor state, the corresponding transition probability must be 1.

state $(2,2) \in S$**:** The state $(2,2)$ corresponds to the PEPA component $F_2 \underset{L}{\bowtie} G_2$. There exist three shared transitions which are enabled and might be executed: Two α-transitions which lead to $(3,3)$, $(4,3)$ respectively, and the β-transition which leads to $(4,3)$. This gives the two possible successor states $(3,3)$ and $(4,3)$. In order to determine the transition probabilities we at first have to determine the apparent rates of the α- and the β-transition in the components F_2 and G_2. These are given by $r_\alpha(F_2) = r_2 + r_3$, $r_\beta(F_2) = r_4$, $r_\alpha(G_2) =$

r_6 and $r_\beta(G_2) = r_7$. Hence, for the three transitions we obtain the rates

$$\begin{aligned}
(2,2) \xrightarrow{\alpha} (3,3): &\quad \frac{r_2 r_6}{r_\alpha(F_2) r_\alpha(G_2)} \min\{r_\alpha(F_2), r_\alpha(G_2)\} = \frac{r_2}{r_2 + r_3} \min\{r_\alpha(F_2), r_\alpha(G_2)\} \\
(2,2) \xrightarrow{\alpha} (4,3): &\quad \frac{r_3 r_6}{r_\alpha(F_2) r_\alpha(G_2)} \min\{r_\alpha(F_2), r_\alpha(G_2)\} = \frac{r_3}{r_2 + r_3} \min\{r_\alpha(F_2), r_\alpha(G_2)\} \\
(2,2) \xrightarrow{\beta} (4,3): &\quad \frac{r_4 r_7}{r_\beta(F_2) r_\beta(G_2)} \min\{r_\beta(F_2), r_\beta(G_2)\} = \min\{r_\beta(F_2), r_\beta(G_2)\}
\end{aligned}$$

To obtain the transition probabilities of the embedded DTMC these rates are scaled such that they sum up to one. It is clear that the two transitions leading from $(2,2)$ to $(4,3)$ must be aggregated into a single one, in order to obtain a proper DTMC.

III.7.1.2 The non-synchronising states

In order to derive local transition probabilities, the local processes must be modified such that the synchronising states are declared as absorbing states. Since in both components the states 1 and 2 are local synchronising states, the first and second row in the generator matrices must be replaced by a zero-vector. The following modified (absorbing) generator matrices are obtained:

Component F:

$$\hat{Q}_F = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -u_1 & 0 & u_1 & 0 \\ 0 & 0 & 0 & -u_2 & 0 & u_2 \\ u_3 & 0 & 0 & 0 & -u_3 & 0 \\ u_4 & 0 & 0 & 0 & 0 & -u_4 \end{pmatrix}$$

Component G:

$$\hat{Q}_G = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -u_5 & u_5 \\ u_7 & u_6 & 0 & -(u_7+u_6) \end{pmatrix}$$

Let Y_F and Y_G be the absorbing CTMCs which are defined by the generator matrices $\hat{Q}_F$ and $\hat{Q}_G$. The transition systems of Y_F and Y_G are shown in Fig. III.11.

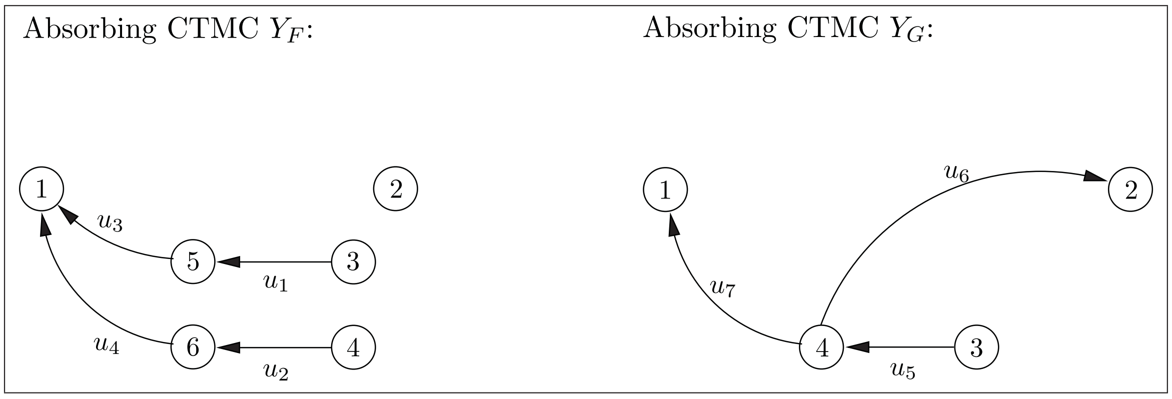


Figure III.11: Running example. The absorbing CTMCs Y_F and Y_G which result from the components F and G by declaring the local synchronising states as absorbing.

state $(3,3) \in NS$**:** If the state 3 (the first component of the global state $(3,3)$) is chosen as the initial state for the absorbing generator matrix $\hat{Q}_F$, it is quickly calculated that absorption will eventually happen in the state 1.

If for the absorbing generator matrix $\hat{Q}_G$ the state 3 (the second component of the global state $(3,3)$) is chosen as the initial state, it is found that absorption will happen in the state 1 with probability $\frac{u_7}{u_7+u_6}$ and in state 2 with probability $\frac{u_6}{u_7+u_6}$.

Hence, we have the following local transition probabilities (for this simple system these probabilities can also be verified by simply looking at the transition systems of the absorbing CTMCs Y_F and Y_G in Fig. III.11):

$$\begin{aligned} \text{in } F: \quad & \mathbb{P}(3 \to 1) = 1. \\ \text{in } G: \quad & \mathbb{P}(3 \to 1) = \frac{u_7}{u_7+u_6}, \\ & \mathbb{P}(3 \to 2) = \frac{u_6}{u_7+u_6}. \end{aligned}$$

By multiplication of the local transition probabilities, the following global transition probabilities are obtained

$$\begin{aligned} \mathbb{P}((3,3) \to (1,1)) &= 1 \cdot \frac{u_7}{u_7+u_6} \\ \mathbb{P}((3,3) \to (1,2)) &= 1 \cdot \frac{u_6}{u_7+u_6}. \end{aligned}$$

state $(2,3) \in NS$**:** If the system is in state $(2,3)$, component F is ready to synchronise but component G is not. As a consequence, component F will stay in state 2 until component G has reached its next synchronising state: component F makes the transition $2 \to 2$ with probability one (convince yourself of the correctness by taking the absorbing state 2 as the initial state of the generator $\hat{Q}_F$). Component G (same results as for the previous state $(3,3)$) performs the transition $3 \to 1$ with probability $\frac{u_7}{u_7+u_6}$, and the transition $3 \to 2$ with probability $\frac{u_6}{u_7+u_6}$. This yields the global transition probabilities

$$\begin{aligned} \mathbb{P}((2,3) \to (2,1)) &= 1 \cdot \frac{u_7}{u_7+u_6} \\ \mathbb{P}((2,3) \to (2,2)) &= 1 \cdot \frac{u_6}{u_7+u_6}. \end{aligned}$$

III.7.2 Example: The Subsystems

In section III.7.1 the set of embedded states was determined. It consists of the following synchronising and non-synchronising states:

$$\{(1,1),(1,2),(2,1),(2,2)\} \in S, \qquad \{(2,3),(3,2),(3,3),(4,2),(4,3)\} \in NS.$$

Suppose we want to determine the steady-state probability of the composite component C in the set $A = \{5,6\} \times \{2,3\} = \{(5,2),(5,3),(6,2),(6,3)\}$. Note, that A does not contain any synchronising states. The first step is to determine $T(\mathbf{x})$ and $T(\mathbf{x})(A)$, for all embedded states $\mathbf{x} \in E^{(X)}$. We must distinguish, whether $\mathbf{x} \in S$ or $\mathbf{x} \in NS$.

Synchronising states. The determination of $T(\mathbf{x})(A)$ is straightforward, for $\mathbf{x} \in S$. Since A does not contain any synchronising states, we have $T(\mathbf{x})(A) = 0$, for all $\mathbf{x} \in S$.

In order to obtain $T((x_1,x_2))$, the PEPA cooperation operator must be applied to x_1 and x_2. This yields the rates of the outgoing (synchronising) transitions. Let $\lambda(\mathbf{x})$ be the sum of all outgoing transitions of $\mathbf{x}$. Then, $1/\lambda(\mathbf{x})$ is the sojourn time in $\mathbf{x}$, i.e., $T((x_1,x_2)) = 1/\lambda(\mathbf{x})$. The following values for $\lambda(\mathbf{x})$ are obtained:

state $\mathbf{x}$	PEPA process	total rate out: $\lambda(\mathbf{x})$
$(1,1)$	$F_1 \underset{L}{\bowtie} G_1$	$\lambda(1,1) = \min\{r_1, r_5\}$
$(1,2)$	$F_1 \underset{L}{\bowtie} G_2$	$\lambda(1,2) = \min\{r_1, r_6\}$
$(2,1)$	$F_2 \underset{L}{\bowtie} G_1$	$\lambda(2,1) = \min\{r_2 + r_3, r_5\}$
$(2,2)$	$F_2 \underset{L}{\bowtie} G_2$	$\lambda(2,2) = \min\{r_2 + r_3, r_6\} + \min\{r_4, r_7\}$

Non-Synchronising states. In order to calculate $T(\mathbf{x})$ and $T(\mathbf{x})(A)$ for non-synchronising states $\mathbf{x} \in NS$, at first, we have to declare the local synchronising states in the components F and G as absorbing states. This was already done in section III.7.1: The resulting absorbing generator matrices were $\hat{Q}_F$ and $\hat{Q}_G$.

Let Y_F be the absorbing CTMC defined by $\hat{Q}_F$ and Y_G be the absorbing CTMC defined by $\hat{Q}_G$. Furthermore, let $Y := (Y_F, Y_G)$ be the joint CTMC. First note, that A does not contain synchronising states: with respect to the absorbing CTMC Y, A is a set of transient states. Now, for each $\mathbf{x} = (x_1, x_2) \in NS$, take $\mathbf{x}$ as the initial state of Y and calculate the mean time to absorption $T(\mathbf{x})$, and the expected total time $T(\mathbf{x})(A)$. This can, e.g., be done in a compositional fashion employing compositional uniformisation.

Fig. III.12 illustrates the scenario, where $(3,3)$ is taken as the initial state of Y (non-reachable states of Y are left out). A comparison of Fig. III.12 with Fig. III.9(b) indicates that Y behaves as the subsystem $SUB((3,3))$, where absorption in Y corresponds to exiting the subsystem $SUB((3,3))$.

III.8 An Application Example

Our approach to decompose PEPA models yields two different types of subsystems. Subsystems which only consist of a synchronising state and subsystems, where the entrance state is a state to be reached immediately after a synchronising transition. Subsystems

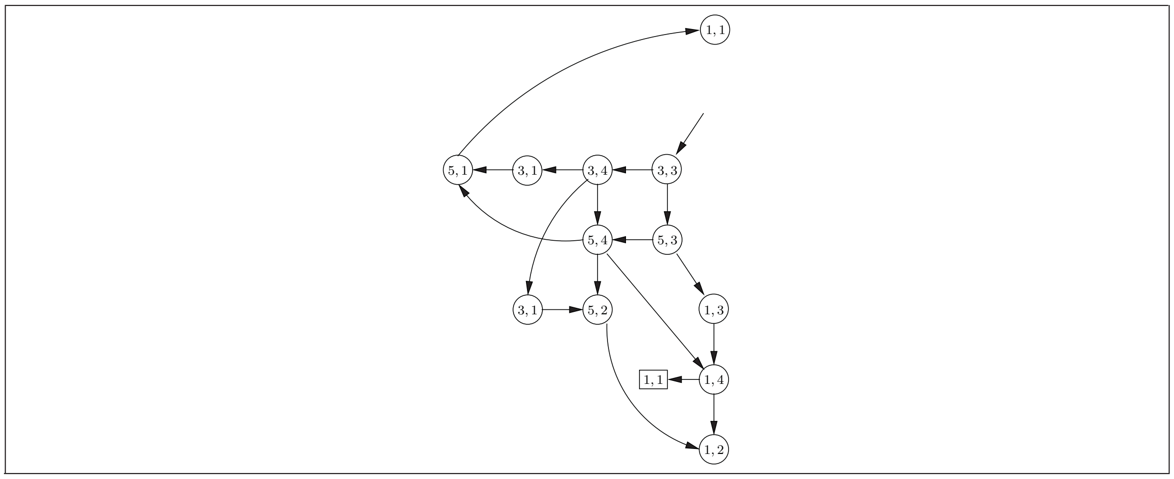


Figure III.12: The CTMC $Y = (Y_F, Y_G)$, with initial state $(3, 3)$ corresponds to the subsystem $SUB((3, 3))$

of the latter type are characterised by a number of parallel processes, where the slowest process, i.e., the process with the greatest individual working time, determines the sojourn time of the entire subsystem. Such a situation frequently arises in Master/Slave architectures, where the master process has to wait for the slowest slave process to complete its work.

In the following we model such a Master/Slave system as a PEPA construct, and afterwards illustrate how this PEPA model can be solved by the approach proposed in this paper. The example is taken from [5], where the author proposes a method to solve Markovian Process Algebra models which is similar to our approach, but less general. We choose the exact same example as in [5] in order to illustrate the similarities and the differences of the two approaches.

III.8.1 Model Description

The composite component *System* which we wish to investigate consists of the parallel composition of m local components: $m - 1$ instances of the component *SlaveSystem* and the component $Master$. These m components carry out global synchronisations over the synchronisation set $L = \{distribute, join\}$.

The component $Master$ distributes a given workload among the $m - 1$ slave systems by executing the activity labelled *distribute*. Afterwards, it is busy with garbage collecting. After that, the master wishes to collect the results from the slave systems, which is indicated by the activity with label *join*. Finally, the master produces new jobs which in the next working cycle are again distributed among the slave systems.

$$\begin{aligned}
System &= (SlaveSystem_1 \underset{L}{\bowtie} SlaveSystem_2 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} SlaveSystem_{m-1}) \underset{L}{\bowtie} Master\\
Master &= (distribute, d).(garbcol, g).(join, j).(newjobs, n).Master\\
SlaveSystem &= Slave \underset{R}{\bowtie} Helper \underset{R}{\bowtie} Helper\\
Slave &= (distribute, d).(working, w).(help, h).(ready, r).(working, w).(join, j).Slave\\
Helper &= (help, h).\big((goodmood, g).(fasthelp, f).(ready, r).Helper\\
&\qquad\qquad + (badmood, b).(slowhelp, s).(ready, r).Helper\big)
\end{aligned}$$

Every slave system *SlaveSystem* is in itself the composition of three components: One component *Slave* and two instances of the component *Helper*. These three components synchronise over the set $R = \{help, ready\}$. The component *Slave* receives a job from the master (via activity labelled *distribute*), then carries out some work, and afterwards requests help from the two helpers (via activity labelled *help*). After that *Slave* waits for both helpers to finish their work, which is followed by some individual work carried out by *Slave*. Finally, *Slave* waits to return the results of its work to the component *Master* via the activity labelled *join*.

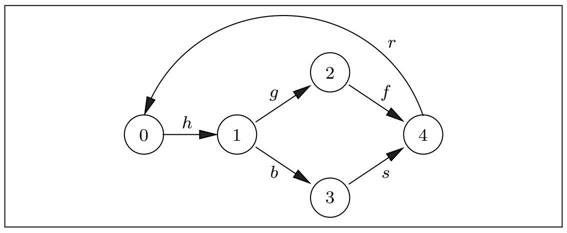
The component *Helper* waits for a help request from the *Slave* (via activity labelled *help*). It then carries out its work and signals the component *Slave* via the activity labelled *ready* that it has finished its work. It is probabilistically chosen whether *Helper* is in good or bad mood, which results in fast or slow performance of *Helper*.

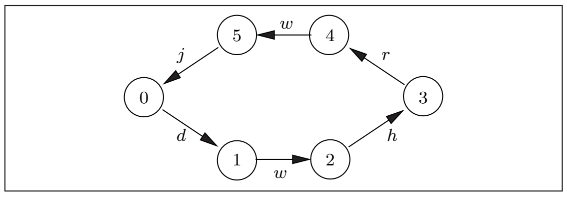
III.8.2 Preliminary Considerations

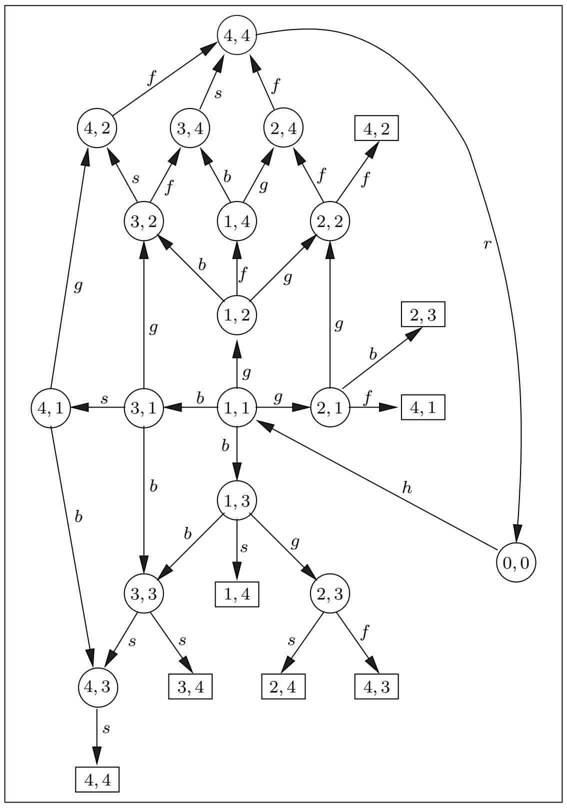
The m components which synchronise over global global synchronising activities, i.e., over activities labelled with action names contained in L, are the $m-1$ components $SlaveSystem_i$, $i = 1 \ldots m$ and the component *Master*. Note that, although the component *SlaveSystem* is itself a composition using the synchronisation set R, the set R does not contribute to the set of embedded states, since a synchronisation produced by an action of R is not global. This in turn means that, in order to apply our proposed approach to solve PEPA models, we must have access to the Markov chains underlying the components $SlaveSystem_i$, $i = 1 \ldots m-1$, and *Master*. Hence, before the actual application of our method we explicitly construct the Markov chains of these components.

The labelled transition system (LTS) of the component *Helper* is shown in Fig III.13(a). The resulting LTS of $Helper \underset{R}{\bowtie} Helper$ is given in Fig. III.13(c). Finally Fig. III.13(d) shows the LTS of the component $SlaveSystem = Slave \underset{R}{\bowtie} Helper \underset{R}{\bowtie} Helper$. These LTSs are immediately obtained by applying the SOS-rules to the involved PEPA components.

For the purpose of simple notation, the states in the LTS of *SlaveSystem* in Fig. III.13(d) are renamed. For example, if in $Slave \underset{R}{\bowtie} Helper \underset{R}{\bowtie} Helper$ all three components are in their local state 0, this would yield the shared state $(0, 0, 0)$. In Fig. III.13(d) this state corresponds to the state 0.


(a) LTS of the component Helper.


(b) LTS of the component Slave.


(c) LTS of Helper $\underset{R}{\bowtie}$ *Helper.*

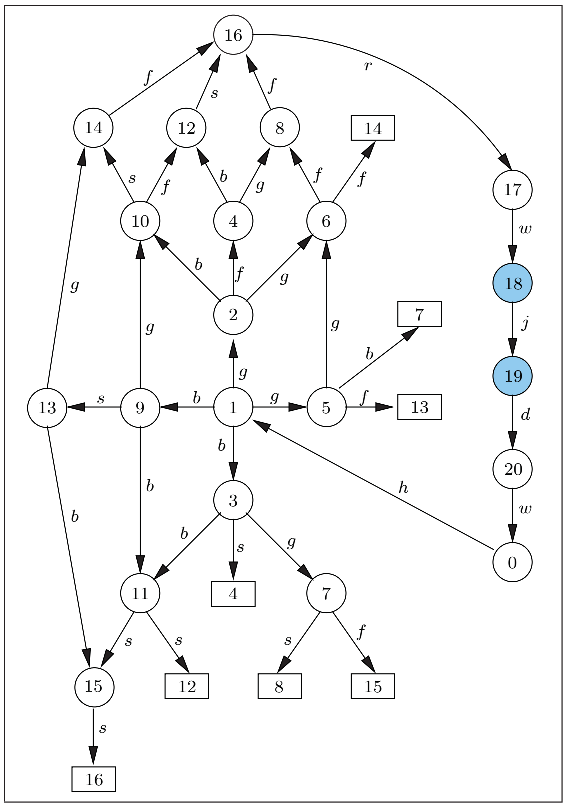

(d) LTS of the component SlaveSystem = Slave $\underset{R}{\bowtie}$ *Helper* $\underset{R}{\bowtie}$ *Helper (note, that the states are renamed). The blue states are local synchronising states (for synchronisation with component Master).*

Figure III.13: *Labelled transition systems.*

From the PEPA description of the component *Master* the corresponding labelled transition system can directly be determinined (see Fig III.14).

The Generator Matrices of the Local Components. It is seen that the generator matrices of the m local components $SlaveSystem_i$, $i = 1 \ldots, m-1$, and *Master* can

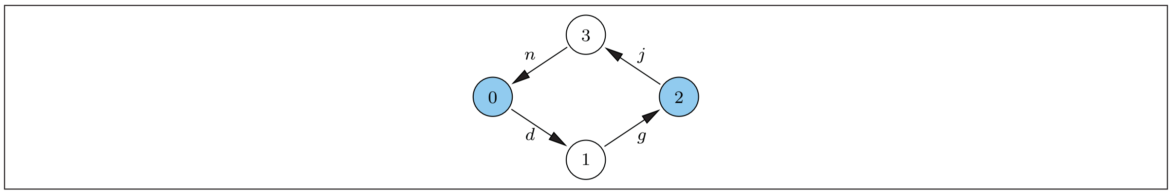


Figure III.14: LTS of $Master$. The blue states are local synchronising states (for synchronisation with component $SlaveSystem$).

easily be determined, e.g., by extracting them from their labelled transition systems in Fig. III.13(d). and Fig. III.14.

Define

- Q_i, $i \in \{1, \ldots, m-1\}$: generator matrix of the CTMC underlying the component $SlaveSystem_i$
- Q_m: generator of the CTMC underlying the component $Master$.

Our algorithm will require knowledge of the modified matrices $\hat{Q}_i$ which result from the Q_i by replacing the rows, associated with local synchronising transitions, by zero row vectors. In the LTS of SlaveSystem we see that the two states 18 and 19 are local synchronising states. That means, $\hat{Q}_i$, $i \in \{1, \ldots, m-1\}$, is obtained from Q_i be replacing the rows 18 and 19 by zero row vectors. In the LTS of the component $Master$ the states 0 and 2 can be identified as local synchronising states. Hence $\hat{Q}_m$ is obtained from Q_m by replacing the rows 0 and 2 by zero row vectors.

In the following solution process of the composite component $System$ we will need the notation

- $\hat{Q}_i$, $i \in \{1, \ldots, m\}$: absorbing generator matrix (obtained by modification of the Q_i)
- Y_i: absorbing Markov chain defined by $\hat{Q}_i$
- $Y := (Y_1, \ldots, Y_m)$: Absorbing joint CTMC of the Y_i.

Measures of Interest. The measure of interest is the steady-state probability that the composite component $System$ is in the set of global states A, where A is required to be the cross-product of sets of local states A_i, i.e., $A = \times_{i=1}^{m} A_i$. The steady-state probability is denoted by $\mathbb{P}(System \in A)$.

We give two examples of which sets A_i one might want to choose and what the interpretation of the resulting steady-state probability is.

- Setting 1: $A_m = \{1\}$, $A_i = \{18\}$, $i = 1 \dots m-1$. In state 1 the component $Master$ performs the garbage collection. In state 18 the slave systems are waiting to synchronise with the master over the transition labelled *join*. That means $\mathbb{P}(System \in A)$ is the probability that all of the slave systems are waiting to synchronise with the component $Master$ while $Master$ is occupied with garbage collection.
- Setting 2: $A_m = \{0, 2\}$, $A_i = \{18, 19\}$, $i = 1 \dots m-1$. The set A_m contains the local synchronising states of the component $Master$, and the other A_i contain the local synchronising states of the slave systems. As a consequence, $A := \times_{i=1}^{m} A_i$ contains all[2] global synchronising states. Hence $\mathbb{P}(System \in A)$ is the probability that $System$ is involved in a (global) synchronisation. This probability might become interesting, when synchronisation proceeds over a communication channel and if the component $System$ is part of a much larger system which uses the same communication channel.

Size of the Model *System* An implementation of the model $System$ in the PEPA workbench revealed the following number of states for different values of m (i.e., $m-1$ is the number of slave systems subordinated to $Master$):

m	3	4	5
number of states	802	16002	320002

As expected, the number of states becomes very large even for small values of m. Hence, computations on the generator matrix of the entire Markov chain underlying the component $System$ become unfeasible, even if smart techniques are used for the storage of the matrix (e.g., Kronecker sums, sparse methods or binary decision diagrams).

III.8.3 Solving the Model *System*

In the following we apply our solution method to the composite component $System$. We anticipate that the algorithm finds 4 embedded states, which results in 4 iteration steps, i.e., 4 calls of the procedure TransitionProbabilities($\cdots$) (p. 31) as well as 4 calls of the procedure ExpectedTotalTimes($\cdots$) (p. 35).

In each iteration step we are given one embedded state $\mathbf{x}$ and we compute the embedded successor states as well as the solution of the subsystem $SUB(\mathbf{x})$ associated with the state $\mathbf{x}$. As for the measure of interest, assume that A_i is a subset of the state space of the component $SlaveSystem_i$, $i = 1, \dots, m-1$, and A_m is a subset of the state space of $Master$. Then, we wish to compute the probability $\mathbb{P}(System \in A)$, where $A := \times_{i=1}^{m} A_i$, and $A_m = \{1\}$, $A_i = \{18\}$, $i = 1 \dots m-1$. This is the measure of setting 1 in the preceding subsection.

[2]Note that A might also contain (global) synchronising states which are not reachable. This, however, does not affect the correctness of our method.

Iteration Step 1. We start with the initial state $(19, \dots, 19, 0)$, i.e., all slave systems are in their local state 19 and the master is in the local state 0. All local states have only the outgoing transition with label *distribute*, hence, this state is an embedded synchronising state, i.e., $(19, \dots, 19, 0) \in S$. Application of the PEPA cooperation operator to all the local states yields the only successor state $(20, \dots, 20, 1)$ and the rate d of the synchronising transition.

- next embedded state: $(20, \dots, 20, 1)$
- transition probability between embedded states: $\mathbb{P}((19, \dots, 19, 0) \to (20, \dots, 20, 1)) = 1$
- $T((19, \dots, 19, 0)) = \frac{1}{d}$
- $T((19, \dots, 19, 0))(A) = 0$, because $(19, \dots, 19, 0) \notin A$

Iteration Step 2. The embedded state $(20, \dots, 20, 1)$ is not a synchronising state, since in the local components there are no synchronising transitions activated. Hence $(20, \dots, 20, 1) \in NS$. According to our algorithm, we must now consider the absorbing joint Markov chain $Y := (Y_1, \dots, Y_m)$, where the generator matrix of the marginal (absorbing) Markov chain Y_i is given by $\hat{Q}_i$. The initial state is, of course $(20, \dots, 20, 1)$, i.e., for each marginal Markov chains $Y_1, \dots, Y_{m-1}$ the initial state is 20, and for Y_m the initial state is 1. The marginal Markov chains $Y_1, \dots Y_{m-1}$ will eventually be absorbed in the state 18 with probability one, and Y_m will eventually be absorbed in state 2 with probability one. That means the next global embedded state is $(18, \dots, 18, 2)$ and the transition probability $\mathbb{P}((20, \dots, 20, 1) \to (18, \dots, 18, 2)) = 1$ (product of the marginal absorption probabilities).

- next embedded state: $(18, \dots, 18, 2)$
- transition probability between embedded states: $\mathbb{P}((20, \dots, 20, 1) \to (18, \dots, 18, 2)) = 1$
- $T((20, \dots, 20, 1))$ is computed as the mean time to absorption of Y, with initial state $(20, \dots, 20, 1)$.
- $T((20, \dots, 20, 1))(A)$ is computed as the expected total time of Y in the set A.

Iteration Step 3. The embedded state $(18, \dots, 18, 2)$ is a synchronising state, since in all local states only synchronising transitions (labelled with the action name *join*) are activated. Hence, we apply the PEPA cooperation operator to the m local states and find that the only successor state is the state $(19, \dots, 19, 3)$, and that the outgoing rate of $(18, \dots, 18, 2)$ is j.

- next embedded state: $(19,\dots,19,3)$
- transition probability between embedded states: $\mathbb{P}((18,\dots,18,2) \rightarrow (19,\dots 19,3)) = 1$
- $T((18,\dots,18,2)) = \frac{1}{j}$
- $T((18,\dots,18,2))(A) = 0$, because $(18,\dots,18,2) \notin A$

Iteration Step 4. The embedded state $(19,\dots,19,3)$ is a non-synchronising state, i.e., $(19,\dots,19,3) \in NS$. Although in the local components $SlaveSystem_i$, $i = 1\dots m-1$, the local state 19 is a synchronising state, in the local component $Master$ the state 3 is non-synchronising, hence, the global state $(19,\dots,19,3)$ is non-synchronising. That means, again we have to solve the absorbing joint Markov chain $Y = (Y_1,\dots,Y_m)$, but this time with the initial state $(19,\dots,19,3)$. For the Y_i, $i = 1\dots m-1$, the marginal initial state 19 is already the absorbing state, i.e., the marginal absorption probabilities in state 19 are one. The Markov chain Y_m, with initial state 3, will eventually become absorbed in the state 0 with probability one. Hence, the only global embedded successor state of $(19,\dots,19,0)$ is the state $(19,\dots,19,0)$. The transition probability, of course, is the product of the marginal absorption probabilities of the Y_i.

- next embedded state: $(19,\dots,19,0)$
- transition probability between embedded states: $\mathbb{P}((19,\dots,19,3) \rightarrow (19,\dots 19,0)) = 1$
- $T((19,\dots,19,3))$ is computed as the mean time to absorption of Y, with initial state $(19,\dots,19,3)$.
- $T((19,\dots,19,3))(A)$ is computed as the expected total time of Y in the set A.

Assembling the Pieces. In the 4 iteration steps, we determined the 4 embedded states $(19,\dots,19,0) \in S$, $(20,\dots,20,1) \in NS$, $(18,\dots,18,2) \in NS$ and $(19,\dots,19,3) \in S$. The set $E^{(X)} := S \cup NS$ determines the discrete time Markov chain X which is embedded in the composite component $Master$. The transition probabilities of the embedded DTMC X were also determined. The resulting discrete time Markov chain is shown in Fig. III.15. Of course, the steady-state distribution of X is given by $\pi = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.

In order to compute the measure of interest $\mathbb{P}(System \in A)$, the only remaining thing to do is to substitute the calculated values from the 4 iteration steps into formula (III.3), i.e., compute

$$\mathbb{P}(System \in A) = \frac{\sum_{\mathbf{x}\in E^{(X)}} \pi(\mathbf{x})T(\mathbf{x})(A)}{\sum_{\mathbf{x}\in E^{(X)}} \pi(\mathbf{x})T(\mathbf{x})}.$$

$$\begin{tikzcd}
 & (19,\ldots,19,3) \arrow[dl, "1"'] & \\
(19,\ldots,19,0) \arrow[dr, "1"'] & & (18,\ldots,18,2) \arrow[ul, "1"'] \\
 & (20,\ldots,20,1) \arrow[ur, "1"'] &
\end{tikzcd}$$

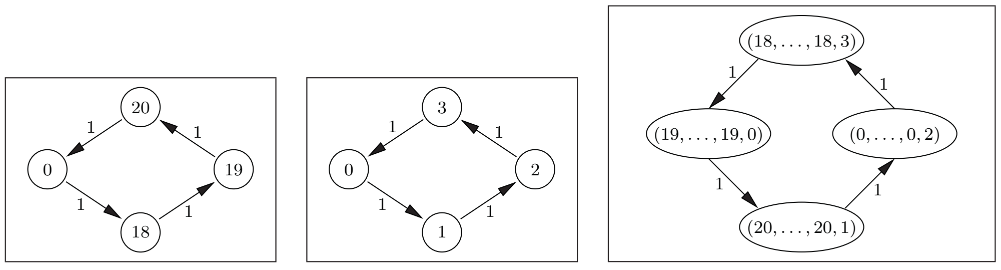
Figure III.15: The embedded DTMC X (embedded in the composite PEPA component *System*).

III.8.4 Comparison to the Bohnenkamp Method

The first thing to note, when comparing our approach to the approach proposed by Bohnenkamp in [5], is that we focus on the MPA language PEPA where synchronisation is always associated with a time consuming activity. In contrast to that, the MPA $\mathscr{YAWN}$ used in [5] relies on the separation of instantaneous (timeless) actions and actions representing an exponential delay, where only instantaneous actions are allowed to synchronise. As a consequence, the CTMC underlying the component *System* modelled in $\mathscr{YAWN}$ does not contain synchronising states in the sense established in this work, i.e., it contains less states. This difference in the number of states is not a property of the solution methods proposed by Bohnenkamp or the current author, but merely a property of the distinct MPA languages.

Construction and Solution of the Embedded DTMC The most severe implication of using these two distinct MPAs is a different one. In $\mathscr{YAWN}$ all probabilistic choices within different sequential components are completely independent of each other. This results in the fact that the embedded DTMC of the composite component is the joint DTMC of independent DTMCs which are embedded in the local components. As a consequence, the steady-state distribution of the DTMC embedded in the composite component can be expressed as the tensor product of local steady-state distributions (belonging to the local embedded DTMCs), i.e., the embedded DTMC can be solved compositionally. In the MPA PEPA the independence of probabilistic choices within different sequential components is not given. To comprehend the following, it is helpful to take a look at Fig. III.13(d) and Fig. III.14. If, for example, the components $SlaveSystem_i$ are in their local embedded state 19 and $Master$ is in its local embedded state 3, then the $SlaveSystem_i$ wait for synchronisation, since $Master$ is not yet ready to synchronise. That means, the next embedded state for the $SlaveSystem_i$ will be 19 again, and for $Master$ the next embedded state will be 0. But if the components $SlaveSystem_i$ are in their local embedded state 19 and $Master$ is in its local embedded state 0, then synchronisation takes place and leads the $SlaveSystem_i$ into their next local embedded state 20 and $Master$ to its next local embedded state 1. In both these cases, the $SlaveSystem_i$ were in their local embedded state 19, but the next local embedded state depended on the state of $Master$.

We could also argue the other way round: At first construct the DTMCs X_i embedded in



(a) DTMCs X_i embedded in SlaveSystem$_i$. *(b) DTMC X_m embedded in Master.* *(c) Irreducibility class of $(X_1, \dots, X_m)$ containing the state $(19, \dots, 19, 0)$.*

Figure III.16: *Embedded local DTMCs X_i and joint DTMC $(X_1, \dots, X_m)$.*

the local components $SlaveSystem_i$, $i = 1 \dots m-1$, and $Master$ (Fig. III.16(a) and Fig. III.16(b)). Afterwards (Fig. III.16(c)) consider the joint DTMC $(X_1, \dots, X_m)$, where we assume that the marginal DTMCs are independent of each other. It is easily seen that neither of the irreducibility classes of $(X_1, \dots, X_m)$, and in particular the irreducibility class containing the initial state $(19, 19, 19, 0)$, coincides with the correct embedded DTMC X shown in Fig. III.15.

Solution of the Subsystems The second difference to the Bohnenkamp method is that we consider computing the expected total time $T(\mathbf{x})(A)$ of the subsystem $SUB(\mathbf{x})$, $x \in E^{(X)}$, and also present algorithms for this purpose. Although the Bohnenkamp method conceptually also decomposes the model under investigation into several subsystems, only the mean sojourn time of these subsystems was considered, hence, the resulting measures were of a much coarser nature. To make this more clear, take the example measure from setting 1 (p. 46). For $A_m = \{1\}$, $A_i = \{18\}$, $i = 1 \dots m-1$, the set $A = \times_{i=1}^{m} = \{(18, \cdots, 18, 1)\}$ contains a single state. $\mathbb{P}(System \in A)$ is that probability that the component $Master$ performs the garbage collection while at the same time all slave systems are waiting to synchronise over the transition with label j. The state $(18, \cdots, 18, 1)$ is reachable only in the subsystem $SUB(20, \dots, 20, 1)$, hence, the steady-state probability $\mathbb{P}(System \in A)$ is given by

$$\mathbb{P}(System \in A) = \frac{\pi((20, \dots, 20, 1))T((20, \dots, 20, 1))(A)}{\sum_{\mathbf{x} \in E^{(X)}} \pi(\mathbf{x})T(\mathbf{x})}.$$

Clearly, the quantity $T((20, \dots, 20, 1))(A)$ is not available within the framework of the Bohnenkamp method, hence, the steady-state probability $\mathbb{P}(System \in A)$ is not computable with that method.

Comparison of Time/Space-Complexity Although, in the Bohnenkamp method the embedded DTMC is efficiently solved by means of a tensor product, whereas in our 2-LDS the embedded DTMC has to be explicitely constructed, the time and space requirements of the two methods, with respect to the solution and storage of the embedded DTMC, can not be compared, because the two methods are applied to different MPAs. If the 2-LDS were transferred to an MPA like $\mathscr{YAWN}$, then the embedded DTMC could be represented as the cross process of DTMCs embedded in the local components just as in the Bohnenkamp method. Within the domain of the 2-LDS applied to PEPA components, however, the embedded DTMC has to be constructed explicitely, hence, the time and storage requirements to solve this DTMC depend on the number of global embedded states which is application dependent. Fortunately, this number is typically small compared to the size of the overall state space of the composite component.

The solution of the subsystems in the 2-LDS differs from the solution of subsystems in the Bohnenkamp method only in the fact that, for each subsystem $SUB(\mathbf{x})$, the 2-LDS computes not only the quantity $T(\mathbf{x})$, but also the additional quantity $T(\mathbf{x})(A)$. Chapter IV will deal with several computation techniques for both quantities. We anticipate that $T(\mathbf{x})(A)$ cannot be obtained as a side-product, when computing $T(\mathbf{x})$, and vice versa. As will also become obvious in Chapter IV, the computation of $T(\mathbf{x})(A)$ does require at most as much time and space as the computation of $T(\mathbf{x})$, for $\mathbf{x} \in E^{(X)}$.

III.9 Conclusion

In this chapter we proposed the novel solution technique 2-LDS (Two-Level Decomposition Scheme) for Markovian Process Algebra Models. It is motivated by the need to antagonise the problem of state space explosion which is inherent to models which are composed of several concurrent components. We chose the MPA language PEPA to illustrate our ideas, which is mainly due to the consistently increasing popularity of this specific MPA. A translation of our ideas to other MPAs, however, should not raise severe obstacles to the interested reader.

Our method is a direct extension of the work in [5] and [6], where the authors develop an algorithm to compute the distribution of a semi-Markov chain which is embedded in the model under investigation. We extend their method by the ability to compute single state probabilities of the original system. Our approach introduces two levels of compositionality:

1. Level: Decomposition of the model into several subsystems.
2. Level: Decomposition of the subsystems.

The most severe restriction of our method is the fact that all concurrent components are only allowed to participate in barrier synchronisations. That means, whenever a global synchronisation takes place, all concurrent components must participate in the synchronisation.

1. Level of Compositionality. In the first level of compositionality the MPA model is decomposed into several subsystems along points of global synchronisation. Global synchronisation points define the set of embedded states $E^{(X)}$ which contains global synchronysing states and global states which immediately succeed a global synchronising state. Every state $\mathbf{x} \in E^{(X)}$ defines a subsystem $SUB(\mathbf{x})$ which is a certain stochastic process subordinated to the original system. We want to clarify what we consider to be a subsystem: As soon as the original system enters the state $\mathbf{x}$ it starts to behave as the subsystem $SUB(\mathbf{x})$. At the next time instant where the system enters an embedded state (a state contained in $E^{(X)}$), say $\mathbf{x}'$, it starts to behave as $SUB(\mathbf{x}')$. All subsystems are *solved* in isolation, with the plan in mind to combine the individual solutions to yield a solution of the entire system. In order to do so, we need additional information about the dynamic between the individual subsystems. This is captured by a discrete time Markov chain (DTMC) which is embedded in the original system. The state space of this embedded DTMC happens to be the set $E^{(X)}$, i.e., each state of the DTMC represents a corresponding subsystem. We summarise the process of computing a solution of the entire system within the first level of compositionality in algorithmic form:

- Find a suitable set of embedded states $E^{(X)}$. This is done by identification of global synchronisation points.
- For each $\mathbf{x} \in E^{(X)}$ solve the subsystem $SUB(\mathbf{x})$.
- Construct and solve the embedded DTMC with state space $E^{(X)}$.
- Combine the individual results.

2. Level of Compositionality. In the second level of compositionality we consider the individual subsystems that are obtained in the first level. Assume that the entire system is the concurrent composition of m components. If the m components were independent of each other, then the system would possess a product-form solution. The components do, however, interact with each other over global synchronisation points, hence, a product-form solution does not exist in general. Fortunately, under certain conditions some of the subsystems can be represented as the parallel (independent) evolution of m absorbing Markov chains. First, a subsystem $SUB(\mathbf{x})$ belongs to one of the following two categories.

- $\mathbf{x}$ is a global synchronising state. In this case the subsystem $SUB(\mathbf{x})$ consists only of this one state and can easily be solved.
- $\mathbf{x}$ is not a synchronising state, but a state to be reached immediately after a global synchronisation.

Obviously, subsystems which belong to the second category are of interest here. Such a subsystem $SUB(\mathbf{x})$ is entered immediately after a global synchronisation has taken place. Once $SUB(\mathbf{x})$ has been entered, the m components start to evolve independently of each other. If one of the components reaches a (local) synchronising state, it stays there until all of the other components also have reached their next local synchronising state. The time instant where the slowest of the m components enters its local synchronising state, is the time instant where the global synchronisation begins; it is also the time instant where the system enters the next subsystem. The point of all this is that the behaviour of $SUB(\mathbf{x})$ can be described by the parallel evolution of m absorbing Markov chains, i.e., by an absorbing joint Markov chain which consists of m marginal absorbing CTMCs. The time instant of leaving $SUB(\mathbf{x})$ coincides with the time instant where the joint CTMC becomes absorbed.

In the subsequent Chapter IV we show that, although the absorbing joint CTMC does not posses a product-form solution, it can be solved efficiently for expected total times of single (global) states. Well-known methods which can be applied will be presented. Furthermore, we will succeed in finding a novel approach to solve the joint CTMC which we refer to as *Compositional Uniformisation* which is able to compete with the known methods with respect to computation time and space requirements.

Chapter IV

Cumulative Measures of Absorbing Joint Markov Chains

In this chapter we derive a relation for the mean time to absorption and the expected total time in some subset of states of an absorbing CTMC Y which is the joint process of independent absorbing CTMCs $Y_1, \ldots, Y_m$. These two quantities can be thought of as cumulative measures, since the mean time to absorption is the cumulated time of Y until absorption, and the expected total time in a subset of the state space is the cumulated time that Y spends in that set before absorption. The relation we will derive exhibits a compositional character in the sense that quantities belonging to the marginal chains are combined to yield the desired cumulative measure.

The straightforward way to compute these cumulative measures of Y would require the explicit construction of the CTMC Y from the marginal components $Y_1, \ldots, Y_m$, i.e., the construction of its generator matrix and the initial probability vector. Such a procedure would have to deal with the problem of state space explosion, since the size of the state space of Y grows exponentially in the number m of marginal CTMCs. The actual computations of the cumulative measures would then be carried out by algorithms which operate on the generator matrix of Y. For a small number m which yields a relatively small state space of Y, the explicit approach would be the method of choice. If, however, the number m results in a huge state space, a method which compositionally computes the cumulative measures of Y from the marginal CTMCs is desirable. The proposition of such a method which we call *compositional uniformisation* is the matter of this chapter. Some of the results presented here were published in [9]. Preliminary considerations that lead to these results can be found in [10].

In section IV.1 the requirements for compositional uniformisation are given. Section IV.2 surveys known methods which compute the mean time to absorption and expected total times of Y. The method of compositional uniformisation is presented in section IV.3, where the proofs are treated separately in section IV.4. In Appendix C a guide of the notation which is used throughout this chapter can be found.

IV.1 The Basic Setup

Let $Y_i = (Y_i(t))_{t \in \mathbb{R}_{\geq 0}}$, $i = 1, \dots, m$, be m independent continuous time homogeneous absorbing Markov chains, where Y_i, $1 \leq i \leq m$, is defined by the finite state space E_i, the starting state $s'_i \in E_i$, the set of absorbing states $S_i \subset E_i$, and the generator matrix $Q_i = (Q_i(j, \ell))_{j,\ell \in E_i}$. With $p_i(t)$ we denote the transient distribution of Y_i at time t.

Now define the Markov chain Y as the joint process $Y = (Y(t))_{t \in \mathbb{R}_{\geq 0}} := (Y_1, \dots, Y_m)$. Then the state space of Y is given by $E = \times_{i=1}^{m} E_i$, the starting state is $s' = (s'_1, \dots, s'_m)$ and the set of absorbing states is given by $S = \times_{i=1}^{m} S_i$. That means the joint Markov chain has reached an absorbing state if all of the marginal Markov chains have become absorbed. It is well-known that the generator matrix Q of the joint CTMC Y can be represented by the Kronecker sum $Q = \oplus_{i=1}^{m} Q_i$. Let $p(t)$ denote the transient distribution of Y at time t.

For $A_i \subset E_i$ and $A := \times_{i=1}^{m} A_i$, with $A \cap S = \emptyset$, i. e. A contains only transient states, we are interested in the mean time to absorption of Y (MTTA) and the expected total time Y spends in A before absorption ($\mathbb{E}_A$).

$$MTTA = \int_0^\infty 1 - p(t)(S)dt \qquad \text{and} \qquad \mathbb{E}_A = \int_0^\infty p(t)(A)dt. \tag{IV.1}$$

IV.2 Well-Known Methods

In this chapter we briefly point out how expected total times of absorbing joint Markov chains can be computed using the known techniques of phase-type distributions and numerical integration.

IV.2.1 Phase-Type Distributions

Every absorbing Markov chain, with a finite state space, describes a phase-type distribution, where each state of the CTMC corresponds to a phase of the phase-type distribution. Hence, in order to derive expected total times of the absorbing CTMC Y, one can rely on the theory of phase-type distributions.

Let Q be the generator matrix of Y and assume that T is a sub-matrix which contains all the transition rates between transient states and be α the initial probability distribution of the transient states. Let H be a random variable for the time to absorption of Y.

Then, the k-th moment of H is given by

$$\mathbb{E}[H^k] = (-1)^k k! \alpha T^{-k} 1, \tag{IV.2}$$

where 1 is a column vector consisting of ones only. The first moment, of course, is the mean time to absorption.

The expected total time $\mathbb{E}_A$ in a transient set A is given by (see [40])

$$\mathbb{E}_A = -\alpha T^{-1} 1_A, \tag{IV.3}$$

where 1_A is a column vector where entries corresponding to the set A are one, and all other entries are zero. It is clear that the dimension of the sub-matrix T is exponential in the number m of parallel CTMCs, hence, computation of the MTTA and $\mathbb{E}_A$ according to (IV.3) suffers from the state space explosion problem.

IV.2.2 Numerical Integration

The total expected time of an absorbing CTMC in a certain subset of the state space can be viewed as the expected accumulated probability mass in that subset before the time instant of absorption. As a consequence, the mean time to absorption of Y and the expected total time Y spends in the set A can be computed by evaluating the integrals from (IV.1):

$$MTTA = \int_0^\infty 1 - p(t)(S)dt \qquad \text{and} \qquad \mathbb{E}_A = \int_0^\infty p(t)(A)dt.$$

The independence of the marginal CTMCs Y_i and the fact that $A = \times_{i=1}^m A_i$, imply the relations $p(t)(S) = \prod_{i=1}^m p_i(t)(S_i)$ and $p(t)(A) = \prod_{i=1}^m p_i(t)(A_i)$. Hence, the integrals above can be written as

$$MTTA = \int_0^\infty 1 - \prod_{i=1}^m p_i(t)(S_i)dt \qquad \text{and} \qquad \mathbb{E}_A = \int_0^\infty \prod_{i=1}^m p_i(t)(A_i)dt.$$

One step in evaluating these integrals numerically is to calculate the integrands at several supporting points, e.g., if $f(t)$ was the integrand then it had to be evaluated for several values of t. Let's take a closer look at the computation of $\mathbb{E}_A$: For every node x, the values $p_1(x)(A_1), \ldots, p_m(x)(A_m)$ must be obtained. For every i, the transient probability $p_i(x)(A_i)$ is obtained by calculating the transient distribution $p_i(x)$ which in turn is given by the matrix exponential

$$p_i(x) = p_i(0)e^{xQ_i}.$$

That means, for every node we have to evaluate m matrix exponentials. If the number of nodes is given by K, we have to evaluate $K \cdot m$ matrix exponentials in order to numerically solve either of the above integrals.

IV.2.3 Uniformisation

Let $P := I + 1/qQ$, where $q \geq \max_j\{|Q(j,j)|\}$, and $\nu(n) := p(0)P^n$. That means $\nu(n) = (\nu(n)(x))_{x \in E}$ is the probability distribution of Y in the n-th step, if uniformised with rate q. The quantity $\nu(n)(\mathcal{A}) := \sum_{x \in \mathcal{A}} \nu(n)(x)$ is the aggregated probability that Y is in $\mathcal{A}$ in the n-th step. Let H be a random variable for the time to absorption.

The transient probability $p(t)(\mathcal{A})$, that Y is in $\mathcal{A}$ at time t, can be expressed by the well-known uniformisation equation (see [7])

$$p(t)(\mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \nu(n)(\mathcal{A}), \tag{IV.4}$$

where $\nu(0) = p(0)$ and, for $n \geq 1$,

$$\nu(n) = p(0)P^n = \nu(n-1)P. \tag{IV.5}$$

Substitution of $p(t)(S)$ and $p(t)(A)$ in (IV.1) yields

$$MTTA = \frac{1}{q} \sum_{n=0}^{\infty} 1 - \nu(n)(S) \quad \text{and} \quad \mathbb{E}_A = \frac{1}{q} \sum_{n=0}^{\infty} \nu(n)(A). \tag{IV.6}$$

In order to obtain the higher moments of the time to absorption H, first consider the following equalities

$$\mathbb{E}[H^k] = \int_0^\infty \mathbb{P}(H^k > x)dx = \int_0^\infty k \cdot t^{k-1} \mathbb{P}(H^k > t^k)dt = \int_0^\infty k \cdot t^{k-1} \mathbb{P}(H > t)dt, \tag{IV.7}$$

where the second equality results from the substitution $x = t^k$. The third equality is a consequence of the equivalence $\{H^k > t^k\} \iff \{H > t\}$, for $H, t \geq 0$.

Now, recall that $p(t)(E \setminus S)$ is the probability that at time t the CTMC Y is not in an absorbing state, i.e., the probability that absorption takes place after time t. According to the uniformisation equation in (IV.4) this yields $\mathbb{P}(H > t) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \nu(n)(E \setminus S)$. Together with equation (IV.7) this yields the following representation of the higher moments of H.

$$\begin{aligned}\mathbb{E}[H^k] &= \int_0^\infty k \cdot t^{k-1} \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \nu(n)(E \setminus S) dt \\ &= \sum_{n=0}^{\infty} \frac{k}{q^k} \frac{(n+k-1)!}{n!} \nu(n)(E \setminus S) \\ &= \sum_{n=0}^{\infty} \frac{k}{q^k} \frac{(n+k-1)!}{n!} (1 - \nu(n)(S)).\end{aligned} \tag{IV.8}$$

Calculating $\nu(n)(S)$ and $\nu(n)(A)$ according to (IV.5) requires a vector-matrix multiplication for every n. Since P is the transition matrix of the joint CTMC $Y = (Y_1, \ldots, Y_m)$, its dimension is exponential in the number m of marginal CTMCs.

Remark 3. One might wonder whether the higher moments of the total times in the set A are just as easily to calculate as the higher moments of the time to absorption in (IV.8). Let H_A be a random variable for the total time which Y spends in the transient set A. Unfortunately, in general the distribution function $\mathbb{P}(H_A \leq t)$ is not as easily to obtain as was the case with $\mathbb{P}(H \leq t)$. If, however, Y is initially in the set A and, once it has left A, never returns to A, then $\mathbb{P}(H_A > t) = p(t)(A)$, or according to (IV.4) $\mathbb{P}(H_A > t) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \nu(n)(A)$. For this special case, we obtain in complete analogy to (IV.8)

$$\mathbb{E}[H_A^k] = \sum_{n=0}^{\infty} \frac{k}{q^k} \frac{(n+k-1)!}{n!} \nu(n)(A) \tag{IV.9}$$

IV.2.4 MeanMax

MeanMax was proposed by Bohnenkamp in [5] as a method to compositionally compute the mean time of the maximum of m phase-type distributed random variables. Since every continuous phase-type distribution corresponds to a suitable absorbing CTMC, this mean value corresponds to the mean value of the maximum of the individual absorption times, or in other words, the mean time to absorption of the absorbing joint CTMC.

In this section, we present the basic idea underlying MeanMax. For $m = 2$, MeanMax works fine. For $m > 2$, however, we show that MeanMax uses a recursion procedure which is based on an erroneous assumption, i.e., MeanMax does not work if more than two absorbing CTMCs are dealt with.

In the following, consider three independent absorbing CTMCs Y_1, Y_2 and Y_3. Let $\alpha(t), \beta(t)$ and $\gamma(t)$ be the corresponding distributions at time t, and set $\alpha := \alpha(0), \beta := \beta(0)$ and $\gamma := \gamma(0)$. Let the parameterised random variable H_ξ^X denote the time to absorption of the CTMC X provided that the initial distribution ξ is used.

MEANMAX for 2 CTMCs. Here, we deal with only the two CTMCs Y_1 and Y_2. The initial distribution of the joint CTMC (Y_1, Y_2) is given by $\alpha \otimes \beta$.

The time to absorption of (Y_1, Y_2) is the maximum of the individual times to absorption of the marginal CTMCs:

$$H_{(\alpha\otimes\beta)}^{(Y_1,Y_2)} = \max\left\{H_\alpha^{Y_1}, H_\beta^{Y_2}\right\} = H_\alpha^{Y_1} + \max\left\{H_\beta^{Y_2} - H_\alpha^{Y_1}, 0\right\}.$$

The right-most maximum term is the length of time that passes between the absorption of Y_1 and the absorption of Y_2. This implies $\max\left\{H_\beta^{Y_2} - H_\alpha^{Y_1}, 0\right\} = H_{\beta(H_\alpha^{Y_1})}^{Y_2}$, and

$$H_{(\alpha\otimes\beta)}^{(Y_1,Y_2)} = H_\alpha^{Y_1} + H_{\beta(H_\alpha^{Y_1})}^{Y_2}.$$

Of course, the expectation of this expression is

$$\mathbb{E}\left[H_{(\alpha\otimes\beta)}^{(Y_1,Y_2)}\right] = \mathbb{E}\left[H_\alpha^{Y_1}\right] + \mathbb{E}\left[H_{\beta(H_\alpha^{Y_1})}^{Y_2}\right].$$

Let

$$\beta' = \mathbb{E}\left[\beta(H_\alpha^{Y_1})\right], \tag{IV.10}$$

i.e., β' is the expected value of the distribution of Y_2 at the time instant $H_\alpha^{Y_1}$, where the CTMC Y_1 becomes absorbed. It is easy to show (see Appendix A) that this implies $\mathbb{E}\left[H_{\beta(H_\alpha^{Y_1})}^{Y_2}\right] = \mathbb{E}\left[H_{\beta'}^{Y_2}\right]$. Consequently, one obtains

$$\mathbb{E}\left[H_{\alpha\otimes\beta}^{Y_1,Y_2}\right] = \mathbb{E}\left[H_\alpha^{Y_1}\right] + \mathbb{E}\left[H_{\beta'}^{Y_2}\right]. \tag{IV.11}$$

The achievement of MeanMax is the discovery of relation (IV.11) and the supply of efficient methods to compute (IV.10). These methods will not be discussed in the current work. In the following, however, we show that MeanMax must fail if it is applied to more than 2 absorbing CTMCs in a recursive fashion.

MEANMAX cannot be applied recursively. What happens if we are dealing with more than two marginal CTMCs? Consider the three absorbing CTMCs Y_1, Y_2 and Y_3, with the initial distributions α, β and γ. We are interested in the mean time to absorption of the joint CTMC (Y_1, Y_2, Y_3) with initial distribution $\alpha \otimes \beta \otimes \gamma$. We can think of (Y_1, Y_2, Y_3) as being composed of the two CTMCs Y_1 and (Y_2, Y_3) with initial distributions α and $\beta \otimes \gamma$. That means we are dealing with $(Y_1, Y_2, Y_3) = (Y_1, (Y_2, Y_3))$ with initial distribution $\alpha \otimes (\beta \otimes \gamma)$. Denote the expected distribution of (Y_2, Y_3), at the time instant where Y_1 becomes absorbed, by

$$(\beta \otimes \gamma)' = \mathbb{E}\left[\beta(H_\alpha^{Y_1}) \otimes \gamma(H_\alpha^{Y_1})\right]. \tag{IV.12}$$

Then the straightforward application of (IV.11) implies that the mean time to absorption of $(Y_1, (Y_2, Y_3))$ is given by

$$\mathbb{E}\left[H^{(Y_1,(Y_2,Y_3))}_{(\alpha\otimes(\beta\otimes\gamma))}\right] = \mathbb{E}\left[H^{Y_1}_{\alpha}\right] + \mathbb{E}\left[H^{(Y_2,Y_3)}_{(\beta\otimes\gamma)'}\right]. \tag{IV.13}$$

In [5] it was wrongly assumed that[1] $(\beta \otimes \gamma)' = \beta' \otimes \gamma'$, where $\beta' = \mathbb{E}\left[\beta(H^{Y_1}_{\alpha})\right]$ and $\gamma' = \mathbb{E}\left[\gamma(H^{Y_1}_{\alpha})\right]$. This lead to the wrong conclusion that in the above equation the right-most term could be replaced by $\mathbb{E}\left[H^{(Y_2,Y_3)}_{(\beta'\otimes\gamma')}\right]$, upon which relation (IV.11) could be applied yet again. As a consequence, MeanMax cannot be applied recursively.

IV.3 Compositional Uniformisation

In section IV.2.3 uniformisation was employed to compute the quantities $\nu(n)(S)$ and $\nu(n)(A)$, which in turn allowed the computation of the moments of the mean time to absorption of Y and expected total times according to formula (IV.6). In the current section we present the new method of compositional uniformisation which computes the quantities $\nu(n)(S)$ and $\nu(n)(A)$ in a compositional way from the marginal CTMCs $Y_1, \dots, Y_m$. That means, it is never necessary to construct or operate on the generator matrix of the joint CTMC Y.

In order to avoid treating $\nu(n)(S)$ and $\nu(n)(A)$ separately, we introduce the new set $\mathcal{A} = \times_{i=1}^{m} \mathcal{A}_i$, where $\mathcal{A}_i \subseteq E_i$, $i = 1 \dots m$. Then, the relations which are derived for $\nu(n)(\mathcal{A})$ are of course also valid for the case of $\mathcal{A} = S$, $\mathcal{A} = A$ respectively.

The following Theorem 2, which is preceded by Definition 6, is the main result of this chapter. Before stating Definition 6 and Theorem 2, we would like to roughly explain their meaning. Definition 6 introduces the $\star$-operator for discrete functions. Now, assume that the CTMCs Y and $Y_1, \dots, Y_m$ are uniformised with suitable rates and let $\nu(n)(\mathcal{A})$ be the probability that $Y \in \mathcal{A}$ in the $n-$th step, and $\nu_i(n)(\mathcal{A}_i)$ be the probability that $Y_i \in \mathcal{A}_i$ in the $n-$th step. Set $\nu_i[\mathcal{A}_i](n) := \nu_i(n)(\mathcal{A}_i)$ to emphasise that $\nu_i(n)(\mathcal{A}_i)$ is considered a discrete function of n. Then, Theorem 2 simply states that $\nu(n)(\mathcal{A}) = (\nu_1[\mathcal{A}_1] \star \cdots \star \nu_m[\mathcal{A}_m])\,(n)$. That means $\nu(n)(\mathcal{A})$ can be calculated by at first computing the functions $\nu_i[\mathcal{A}_i]$ from the marginal CTMCs, and afterwards combining these functions via the $\star-$operator.

Definition 6. For two discrete functions $f, g : \mathbb{N}_0 \to [0,1] \subset \mathbb{R}$ and constant values

[1] Note that the distribution of (Y_2, Y_3) at time t is given by $\beta(t) \otimes \gamma(t)$. In contrast to that, the expected distribution of (Y_2, Y_3) at a random time H is given by $\mathbb{E}[\beta(H) \otimes \gamma(H)]$ which in general is not equal to $\mathbb{E}[\beta(H)] \otimes \mathbb{E}[\gamma(H)]$.

$q_f, q_g \in \mathbb{R}_{>0}$ assigned to these functions, define the $\star$-operator by

$$(f \star g)(n) = \sum_{k=0}^{n} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f(k)g(n-k) \qquad \text{and} \qquad q_{f\star g} = q_f + q_g. \quad \text{(IV.14)}$$

Also let $\star_{i=1}^m f_i := f_1 \star \cdots \star f_m$.

$\square$

An intuitive meaning of the $\star$-operator is provided in the subsequent Section IV.3.1.

Theorem 2. *For $i = 1 \dots m$, let $P_i := I + 1/q_i Q_i$ and $v_i(0) := p_i(0)$, $\nu_i(n) = \nu_i(n-1)P_i$, for $n \geq 1$, where $q_i \geq \max_j\{|Q_i(j,j)|\}$.*

Then, with $\nu_i[\mathcal{A}_i](n) = \nu_i(n)(\mathcal{A}_i)$, $i = 1 \dots m$, and $q := q_1 + \cdots + q_m$, we have

(i) $p(t)(\mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$,

(ii) We also have

$$\left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n) = \nu(n)(\mathcal{A}), \quad \text{(IV.15)}$$

i.e., $\left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$ is the probability that Y, if uniformised with rate q, is in $\mathcal{A}$ in the $n-$th step.

Proof. In Section IV.4. $\square$

The definition of the $\star$-operator indicates the possibility of an approximation scheme for the values $\nu(n)(\mathcal{A})$. Note the involvement of the binomial distribution in the sum in (IV.14). In the following we investigate the quality of an approximation to $\nu(n)(\mathcal{A})$ which results from cutting the lower and upper tail of this binomial distribution.

At first, we define a modified version of the $\star$-operator in the following Definition 7, namely the $|\star|$-operator. With this modified operator the value $\left(|\star|_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$ will be an approximation to $\left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$.

Definition 7. Under the same prerequisites as in Definition 6 and for $0 \leq \ell < r \leq n$ define

$$(f \left|\overset{r}{\underset{\ell}{\star}}\right| g)(n) = \sum_{k=\ell}^{r} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f(k)g(n-k) \qquad \text{and} \qquad q_{f|\star_\ell^r|g} = q_f + q_g.$$

The indices ℓ and r are cutoff indices. For the general case of m functions we will also use the abbreviation

$$\left|\overset{\mathbb{r}}{\underset{\mathbb{l}}{\star}}\right|_{i=1}^{m} f_i = f_1 \left|\overset{r_1}{\underset{\ell_1}{\star}}\right| f_2 \left|\overset{r_2}{\underset{\ell_2}{\star}}\right| \cdots \left|\overset{r_{m-1}}{\underset{\ell_{m-1}}{\star}}\right| f_m,$$

where $\mathbb{l} = (\ell_1, \dots, \ell_{m-1})$ and $\mathbb{r} = (r_1, \dots, r_{m-1})$ are multi-indices.

□

In Theorem 2 the values $\nu(n)(\mathcal{A})$ were expressed via the $\star$-operator. The following Theorem 3 considers the error of approximations $\widetilde{\nu}(n)(\mathcal{A})$ which result by simply replacing the $\star$-operator by the $|\star|$-operator.

Theorem 3. *Under the same prerequisites as in Theorem 2 Let* $\mathbb{l} = (\ell_1, \ldots, \ell_{m-1})$ *and* $\mathbb{r} = (r_1, \ldots, r_{m-1})$ *be multi-indices such that, for* $i = 1 \ldots m-1$ *and a given* $0 \le \gamma < 1$,

$$1 - \sum_{k=\ell_i}^{r_i} \binom{n}{k} \frac{(\sum_{j=1}^{i} q_j)^k q_{i+1}^{n-k}}{(\sum_{j=1}^{i+1} q_j)^n} \le \gamma,$$

i.e., l_i *and* r_i *are cutoff indices for the binomial distribution such that the cut off probability mass does not exceed* γ.

Let

$$\widetilde{\nu}(n)(\mathcal{A}) := (\left|\overset{\mathbb{r}}{\underset{\mathbb{l}}{\star}}\right|_{i=1}^{m} \nu_i[\mathcal{A}_i])(n).$$

Then the following holds

$$0 \le \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \le 1 - (1-\gamma)^{m-1}.$$

Proof. In section IV.4. □

Remark 4. In the above theorem, we cut off the involved binomial distributions at given cut-off indices and afterwards calculate the resulting error of the approximate value $\widetilde{\nu}(n)(\mathcal{A})$. In practise it is more convenient to proceed in the opposite order. Let $0 \le \Gamma < 1$. Then, in order to ensure

$$0 \le \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \le \Gamma,$$

we have to choose $\gamma := 1 - \sqrt[m-1]{1-\Gamma}$ in the above Theorem 3.

IV.3.1 Probabilistic Interpretation

In order to understand the structure of the $\star$-operator consider the example $Y = (Y_1, Y_2)$, where Y_1 has uniformisation rate q_1 and Y_2 has uniformisation rate q_2. The sequence of steps of (Y_1, Y_2) is the superposition of two individual Poisson streams and has rate $q_1 + q_2$. Then, the following two facts (partly illustrated in Fig. IV.1), are clear:

(a) The event that out of n steps of (Y_1, Y_2), Y_1 has contributed k steps and Y_2 has contributed $n-k$ steps happens with the binomial probability $\binom{n}{k}\left(\frac{q_1}{q_1+q_2}\right)^k \left(\frac{q_2}{q_1+q_2}\right)^{n-k}$.

(b) $\nu_1(\mathcal{A}_1)(k)\nu_2(\mathcal{A}_2)(n-k)$ is the probability that $Y \in \mathcal{A}_1 \times \mathcal{A}_2$, provided that Y_1 has performed k steps and Y_2 has performed the other $n-k$ steps.

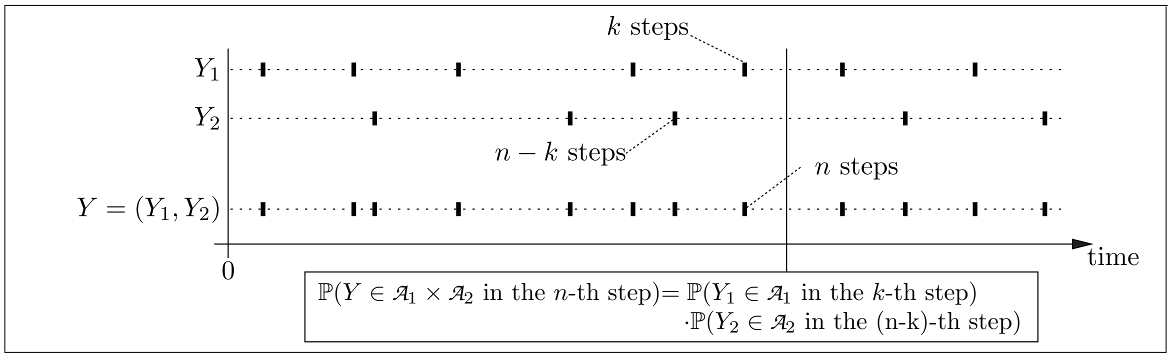

k steps
Y_1
Y_2
$n-k$ steps
n steps
$Y=(Y_1,Y_2)$
0
time
$\mathbb{P}(Y \in \mathcal{A}_1 \times \mathcal{A}_2$ in the n-th step)= $\mathbb{P}(Y_1 \in \mathcal{A}_1$ in the k-th step)
$\cdot\mathbb{P}(Y_2 \in \mathcal{A}_2$ in the (n-k)-th step)

Figure IV.1: Marginal and joint steps and probabilities.

By the law of total probability the following sum is the (unconditional) probability that $Y \in \mathcal{A}_1 \times \mathcal{A}_2$ in the n-th step.

$$(\nu_1 \star \nu_2)(n) = \sum_{k=0}^{n} \binom{n}{k} \left(\frac{q_1}{q_1+q_2}\right)^k \left(\frac{q_2}{q_1+q_2}\right)^{n-k} \nu_1(\mathcal{A}_1)(k)\nu_2(\mathcal{A}_2)(n-k).$$

It is clear that this line of argumentation can be generalised to more than two parallel CTMCs. On the one hand this explaines the definition of the $\star$-operator and on the other hand we have just derived statement (ii) of Theorem 2 (the experienced reader will immediately note that in Theorem 2 statement (ii) implies statement (i)).

IV.3.2 CU Algorithms and Complexity

The original goal of this chapter was to compute the moments of the time to absorption of Y and the expected total time of Y in the set A, i.e., the computation of the sums in (IV.8) and (IV.6) which are replicated hereafter:

$$\mathbb{E}[H^k] = \sum_{n=0}^{\infty} \frac{k}{q^k} \frac{(n+k-1)!}{n!}(1-\nu(n)(S)) \qquad \text{and} \qquad \mathbb{E}_A = \frac{1}{q}\sum_{n=0}^{\infty} \nu(n)(A). \tag{IV.16}$$

For $k=1$, $\mathbb{E}[H^k]$ is the mean time to absorption of Y. For actual computations, the sums must be truncated at a certain index. Let N be that truncation index.

The following algorithms compute the values $\left(\star_{i=1}^{m}\nu_i[\mathcal{A}_i]\right)(n) = \nu(n)(\mathcal{A})$, for $n = 0 \dots N$, and $\mathcal{A} = \times_{i=1}^{m}\mathcal{A}_i$. $\mathcal{A}$ might be seen as a placeholder for S and A. The term $b(k;n,p)$ denotes the binomial probability $b(k;n,p) = \binom{n}{k}p^k(1-p)^{n-k}$, and with $b(\cdot;n,p)$ we denote the entire binomial distribution with parameters n and p. In Appendix B it is shown how these binomial probabilities can be computed.

For a given number N, the first algorithm (ExactCU) computes the exact values $\nu(n)(\mathcal{A})$,

$n = 1 \dots N$. The second algorithm (ApproximateCU) needs an additional error parameter $\gamma > 0$, and computes the approximate values $\widetilde{\nu}(n)(\mathcal{A})$, $n = 0 \dots N$, with $0 \leq \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \leq 1 - (1-\gamma)^{m-1}$.

After execution of the algorithm ExactCU, ApproximateCU respectively, the obtained values $\nu(n)(\mathcal{A})$, $\widetilde{\nu}(n)(\mathcal{A})$ respectively, $n = 0 \dots N$, can be substituted into the formulas in (IV.16).

```
Procedure ExactCU($\mathcal{A}_1, \dots, \mathcal{A}_m, \nu_1[\mathcal{A}_1], \dots, \nu_m[\mathcal{A}_m], N$)
for $n = 0 \dots N$ do
    for $i = 1 \dots m$ do
        $\nu_i[\mathcal{A}_i](n) = \sum_{x \in \mathcal{A}_i} \nu_i(n)(x)$;          /* (a) $O(d_i)$ */
        $\nu_i(n+1) = \nu_i(n) P_i$;                                /* (b) $O(d_i^2)$ */
    /* compute $f_m(n) = (\star_{k=1}^m \nu_k[\mathcal{A}_k])(n) = \nu(n)(\mathcal{A})$ iteratively: */
    $f_1(n) := \nu_1[\mathcal{A}_1](n)$;
    for $i = 2 \dots m$ do
        $1 - p = q_i / \sum_{j=1}^i q_j$;
        compute $b(\cdot; n, p)$;                                   /* (c) $O(n)$ */
        $f_i(n) = (f_{i-1} \star \nu_i[\mathcal{A}_i])(n)$;             /* (c) $O(n)$ */
        $q_{f_i} = q_1 + \dots + q_i$;
        store $f_i(n)$;
    output $f_m(n) = \nu(n)(\mathcal{A})$;
```

```
Procedure ApproximateCU($\mathcal{A}_1, \dots, \mathcal{A}_m, \nu_1[\mathcal{A}_1], \dots, \nu_m[\mathcal{A}_m], N, \gamma$)
for $n = 0 \dots N$ do
    for $i = 1 \dots m$ do
        $\nu_i[\mathcal{A}_i](n) = \sum_{x \in \mathcal{A}_i} \nu_i(n)(x)$;          /* (a) $O(d_i)$ */
        $\nu_i(n+1) = \nu_i(n) P_i$;                                /* (b) $O(d_i^2)$ */
    /* compute $\widetilde{f}_m(n) = \left(|\star_{\mathbb{l}}^{\mathbb{r}}|_{k=1}^m \nu_k[\mathcal{A}_k]\right)(n) = \widetilde{\nu}(n)(\mathcal{A})$ iteratively: */
    $\widetilde{f}_1(n) := \nu_1[\mathcal{A}_1](n)$;
    for $i = 2 \dots m$ do
        $1 - p = q_i / \sum_{j=1}^i q_j$;
        find $\ell_n, r_n$, with $1 - \sum_{k=\ell_n}^{r_n} b(k; n, p) \leq \gamma$;          /* $O(1)$ */
        for $k = \ell_n \dots r_n$ do                              /* (c) $O(\sqrt{n})$ */
            compute $b(k; n, p)$;
        $\widetilde{f}_i(n) = \left(\widetilde{f}_{i-1} \,|\star_{\ell_n}^{r_n}|\, \nu_i[\mathcal{A}_i]\right)(n)$;       /* (c) $O(\sqrt{n})$ */
        $q_{\widetilde{f}_i} = q_1 + \dots + q_i$;
        store $\widetilde{f}_i(n)$;
    output $\widetilde{f}_m(n) = \widetilde{\nu}(n)(\mathcal{A})$;
```

Let $d_i = dim(P_i)$, $i = 1 \ldots m$. The storage requirements for the matrices P_i and the vectors $\nu_i(n)$ and $\nu_i(n-1)$, $i = 1 \ldots m$, are $O\left(\sum_{i=1}^m d_i^2\right)$ for every $n > 0$. The values $f_i(n)$, $n = 0 \ldots N$, $i = 1 \ldots m$, must be stored permanently, which requires space $O(mN)$. Hence, the overall storage requirement of the algorithm lies in $O\left(\sum_{i=1}^m d_i^2 + mN\right)$.

The time complexities of the most time-consuming statements are given as comments in the algorithms. In line 8 of the procedure ApproximateCU($\cdot$) truncation indices ℓ_n and r_n for the binomial distribution are computed. This computation proceeds in time $O(1)$ and yields $r_n - \ell_n \in O(\sqrt{n})$ as is explained in Appendix B, where the computation of binomial probabilities is treated. For every $n \in \{0, \ldots, N\}$, the number of operations in the above algorithms are:

$$\begin{aligned}
\textbf{exact CU: } &(a)\ \ O\left(\sum_{i=1}^m d_i\right), \quad (b)\ \ O\left(\sum_{i=1}^m d_i^2\right), \quad (c)\ \ O(mn). \\
\textbf{approximate CU: } &(a)\ \ O\left(\sum_{i=1}^m d_i\right), \quad (b)\ \ O\left(\sum_{i=1}^m d_i^2\right), \quad (c)\ \ O(m\sqrt{n}).
\end{aligned}$$

After N iteration steps, we arrive at the overall time complexities

$$\begin{aligned}
\textbf{exact CU: } O\left(N\sum_{i=1}^m d_i^2 + mN^2\right) &= O\left(N\left(\sum_{i=1}^m d_i^2 + mN\right)\right). \\
\textbf{approximate CU: } O\left(N\sum_{i=1}^m d_i^2 + mN^{3/2}\right) &= O\left(N\left(\sum_{i=1}^m d_i^2 + m\sqrt{N}\right)\right).
\end{aligned} \tag{IV.17}$$

IV.4 Proofs of the Theorems 2 and 3

IV.4.1 Proof of Theorem 2

Before the proof of Theorem 2 consider the following lemma.

Lemma 4. *With the functions $a_i, 1 \leq i \leq m$, and the values $q_{a_i} = q_i$ assigned to them, the following assertion holds*

$$\left(\star_{i=1}^m a_i\right)(n) = \sum_{\substack{n_i \geq 0, \\ \sum_i n_i = n}} n! \prod_{i=1}^m \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! q^{n_i}}\right].$$

Proof. We proof this by induction over m. For $m = 1$ the assertion is trivially true and for $m = 2$ it is easily verified that

$$\left(\star_{i=1}^{m} a_i\right)(k) = \sum_{\substack{n_i \geq 0,\\ \sum_{i=1}^{m} n_i = k}} k! \prod_{i=1}^{m} \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! \left(\sum_1^m q_\ell\right)^{n_i}} \right],$$

which provides us with a valid induction hypothesis. For the induction step we write

$$(a_1 \star a_2 \star \cdots \star a_{m+1})(n) = \sum_{k=0}^{n} \binom{n}{k} \frac{\left(\sum_1^m q_i\right)^k q_{m+1}^{n-k}}{\left(\sum_i^{m+1} q_i\right)^n} \left(\star_1^m a_i\right)(k) a_{m+1}(n-k). \qquad \text{(IV.18)}$$

Now substitute $\left(\star_1^m a_i\right)(k)$ in (IV.18) with the induction hypothesis and afterwards combine the resulting two sums using the substitution $n - k = n_{m+1}$:

$$(a_1 \star a_2 \star \cdots \star a_{m+1})(n) = \sum_{\substack{n_i \geq 0,\\ \sum_{i=1}^{m+1} = n}} \binom{n}{n - n_{m+1}} \frac{\left(\sum_1^m q_i\right)^{n-n_{m+1}} q_{m+1}^{n_{m+1}}}{\left(\sum_i^{m+1} q_i\right)^n}$$
$$\cdot (n - n_{m+1})! \prod_{i=1}^{m} \left[\frac{q_i^{n_i} a_i(n_i)}{n_i! \left(\sum_1^m q_\ell\right)^{n_i}} \right] a_{m+1}(n_{m+1}).$$

Extending the range of the product sign to $m+1$ and proper rearrangement of the terms yields the assertion. □

Assertion of Theorem 2. *For $i = 1 \ldots m$, let $P_i := I + 1/q_i Q_i$ and $v_i(0) := p_i(0)$, $\nu_i(n) = \nu_i(n-1)P_i$, for $n \geq 1$, where $q_i \geq \max_j\{|Q_i(j,j)|\}$.*

Then, with $\nu_i[\mathcal{A}_i](n) = \nu_i(n)(\mathcal{A}_i)$, $i = 1 \ldots m$, and $q := q_1 + \cdots + q_m$, we have

(i) $p(t)(\mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)$,

(ii) With $\nu(n)$ from (IV.5), we have

$$\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n) = \nu(n)(\mathcal{A}), \qquad \text{(IV.19)}$$

i.e., $\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)$ is the probability that Y, if uniformised with rate q, is in $\mathcal{A}$ in the $n-$th step.

Proof.
Proof of statement (i): By independence of the marginal CTMCs

$$p(t)(\mathcal{A}) = p_1(t)(\mathcal{A}_1) \cdots p_m(t)(\mathcal{A}_m). \qquad \text{(IV.20)}$$

For a given initial distribution $p_i(0)$, some $q_i \geq \max_{j \in E_i}\{|Q_i(j,j)|\}$ and with $P_i := I +$

$1/q_i Q_i$, the probability $p_i(t)(\mathcal{A}_i)$ can be expressed as the series

$$p_i(t)(\mathcal{A}_i) = \sum_{k=0}^{\infty} \frac{(q_i t)^k}{k!} e^{-q_i t} \nu_i(k)(\mathcal{A}_i), \tag{IV.21}$$

where $\nu_i(0) = p_i(0)$ and $\nu_i(k+1) = \nu_i(k)P_i$, for $k \geq 0$. Throughout this section let

$$\begin{aligned} q &:= q_1 + \cdots + q_m, \\ a_i(n) &:= \nu_i(n)(\mathcal{A}_i), \text{ for } i = 1 \ldots m, n \geq 0. \end{aligned}$$

Substituting each $p_i(\mathcal{A}_i)$ in (IV.20) with the corresponding series in (IV.21) yields

$$\begin{aligned} p(t)(\mathcal{A}) &= \prod_{i=1}^{m} \sum_{n_i=0}^{\infty} \frac{(q_i t)^{n_i}}{n_i!} e^{-q_i t} a_i(n_i) \\ &= e^{-qt} \prod_{i=1}^{m} \sum_{n_i=0}^{\infty} \frac{(q_i t)^{n_i}}{n_i!} a_i(n_i) \\ &= e^{-qt} \sum_{n=0}^{\infty} \sum_{\substack{n_i \geq 0, \\ \sum_i n_i = n}} \prod_{i=1}^{m} \frac{(q_i t)^{n_i}}{n_i!} a_i(n_i) \\ &= \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \cdot \sum_{\substack{n_i \geq 0, \\ \sum_i n_i = n}} \left[n! \prod_{i=1}^{m} \frac{q_i^{n_i} a_i(n_i)}{n_i! q^{n_i}} \right]. \end{aligned} \tag{IV.22}$$

Equation (IV.22) in combination with lemma 4 proves statement (i).

Proof of statement (ii): Note first that from $q_i \geq \max_j\{|Q_i(j,j)|\}$, $i = 1 \ldots m$, follows $q = \sum_i q_i \geq \max_j\{|Q(j,j)|\}$, which is a consequence of $Q = \oplus_{i=1}^{m} Q_i$. Now, on the one hand $p(t)(\mathcal{A})$ can be expressed by the series in (i) and on the other hand by the uniformisation equation in (IV.4). Equating the right-hand side expressions of (IV.4) and of (i) it follows that, for all $t \geq 0$,

$$\sum_{n=0}^{\infty} \frac{(qt)^n}{n!} \left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} \nu(n)(\mathcal{A}) < \infty, \tag{IV.23}$$

where convergence is assured by the fact that $\nu(n)(\mathcal{A}) \in [0,1]$ is a probability. The uniqueness theorem for power series states that if for two series $\sum a_n x^n$ and $\sum b_n x^n$ with positive convergence radii there exists a sequence $(g_k)_{k \geq 0}$, with $0 \neq g_k \to 0$, such that $\sum a_n g_k^n = \sum b_n g_k^n$, for all k, then $a_n = b_n$, for $n \geq 0$. Since (IV.23) holds for all $t \geq 0$, such a sequence obviously exists for both power series in (IV.23), hence, $\frac{\nu(n)(\mathcal{A})}{n!} = \frac{\left(\star_{i=1}^{m} \nu_i[\mathcal{A}_i]\right)(n)}{n!}$. This implies the assertion. □

IV.4.2 Proof of Theorem 3

In order to proof Theorem 3 we prepend the following lemma.

Lemma 5. *Consider the same prerequisites as in Definition 7. In addition let* $\gamma_1, \gamma_2 \in \mathbb{R}$, *with* $0 \leq \gamma_1, \gamma_2 < 1$. *Let* $\ell, r \in \mathbb{N}_0$ *be such that*

$$1 - \sum_{k=\ell}^{r} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} \leq \gamma_1, \tag{IV.24}$$

and let $f' : \mathbb{N}_0 \to [0,1] \subset \mathbb{R}$ *be a discrete function, with*

$$0 \leq f(k) - f'(k) \leq \gamma_2, \quad \text{for } 0 \leq k \leq n,$$

Then, the following is true

$$0 \leq (f \star g)(n) - (f' \left|\overset{r}{\underset{\ell}{\star}}\right| g)(n) \leq \gamma_2(1-\gamma_1) + \gamma_1.$$

Proof. It is clear that $(f \star g)(n) - (f' \left|\star_\ell^r\right| g)(n) \geq 0$. To proof the upper bound, consider

$$\begin{aligned}(f \star g)(n) = & \overbrace{\sum_{k=\ell}^{r} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f'(k) g(n-k)}^{=(f'\left|\star_\ell^r\right|g)(n)} \\ & + \sum_{k=\ell}^{r} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} (f(k) - f'(k)) g(n-k) \\ & + \sum_{k=0}^{\ell-1} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f(k) g(n-k) \\ & + \sum_{k=r+1}^{n} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n} f(k) g(n-k).\end{aligned}$$

The bounds $f, g \leq 1$ and $f(k) - f'(k) \leq \gamma_2$, for $0 \leq k \leq n$, and the fact that the $\binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n}$ are binomial probabilities, imply

$$(f \star g)(n) \leq (f' \left|\overset{r}{\underset{\ell}{\star}}\right| g)(n) + \gamma_2(1-\gamma_1) + 1 - \sum_{k=\ell}^{r} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f + q_g)^n}.$$

Together with (IV.24) this concludes the proof. □

Assertion of Theorem 3. *Under the same prerequisites as in Theorem 2 Let* $\mathbb{l} = (\ell_1, \ldots, \ell_{m-1})$ *and* $\mathbb{r} = (r_1, \ldots, r_{m-1})$ *be multi-indices such that, for* $i = 1 \ldots m-1$ *and a* *given* $0 \le \gamma < 1$,

$$1 - \sum_{k=\ell_i}^{r_i} \binom{n}{k} \frac{(\sum_{j=1}^{i} q_j)^k q_{i+1}^{n-k}}{(\sum_{j=1}^{i+1} q_j)^n} \le \gamma,$$

and let

$$\widetilde{\nu}(n)(\mathcal{A}) := (\left|\overset{\mathbb{r}}{\underset{\mathbb{l}}{\star}}\right|_{i=1}^{m} \nu_i[\mathcal{A}_i])(n).$$

Then the following holds

$$0 \le \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \le 1 - (1-\gamma)^{m-1}.$$

Proof. We proof this by induction over m. Take $\mathbb{f} = \mathbb{f}' := \nu_1[\mathcal{A}_1]$ and $\mathbb{g} := \nu_2[\mathcal{A}_2]$. Then corollary 5 implies $0 \le (\mathbb{f} \star \mathbb{g})(n) - (\mathbb{f}' \left|\star_{\ell_1}^{r_1}\right| \mathbb{g}) \le \gamma$, which is the statement of the current theorem for $m = 2$. That means the following induction hypothesis holds:

$$0 \le (\star_{i=1}^{m} \nu_i[\mathcal{A}_i])(n) - (\left|\overset{\mathbb{r}}{\underset{\mathbb{l}}{\star}}\right|_{i=1}^{m} \nu_i[\mathcal{A}_i])(n) \le 1 - (1-\gamma)^{m-1}.$$

For the induction step take $f := (\star_{i=1}^{m} \nu_i[\mathcal{A}_i])$, $g := \nu_{m+1}[\mathcal{A}_{m+1}]$ and $f' := |\star_{\mathbb{l}}^{\mathbb{r}}|_{i=1}^{m} \nu_i[\mathcal{A}_i]$. That means, it remains to prove

$$0 \le (f \star g)(n) - (f' \left|\overset{r_{m+1}}{\underset{\ell_{m+1}}{\star}}\right| g)(n) \le 1 - (1-\gamma)^{m}.$$

Since by induction hypothesis $0 \le f(n) - f'(n) \le 1 - (1-\gamma)^{m-1}$, corollary 5 verifies this inequality. □

IV.5 CU: Two Generic Numerical Examples

IV.5.1 Example Cases

Consider an absorbing joint CTMC $Y = (Y_1, \dots, Y_m)$ in the context of two examples. The generator matrices of the marginal CTMCS Y_i are given by

example 1:
$$Q_i = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 \\ 0 & 0 & -3 & 3 & 0 \\ 4 & 0 & 0 & -8 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

example 2:
$$Q_i = \begin{pmatrix} -7 & 1 & 2 & 3 & 1 \\ 2 & -11 & 3 & 4 & 2 \\ 3 & 4 & -15 & 5 & 3 \\ 4 & 5 & 6 & -19 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and in both examples the initial state of each marginal CTMC is the state 1. Of course, the generator matrix of Y is given by $Q := \oplus_{i=1}^{m} Q_i$, and the set of absorbing states of Y is denoted by S. For both examples define the set of considered transient states by $A_i := \{1, 3\}$ and $A := \times_{i=1}^{m} A_i$. That means $\mathbb{E}_A$ is the expected total time, where all the marginal CTMCs are in state 1 or 3 at the same time. We remark that in all of our computations we did not exploit the fact that the generator matrices are identical for all marginal CTMCs.

Interpretation of Example 1 Consider a system consisting of m parallel machines, where the i-th machine is modelled by the CTMC Y_i. Assume that each machine has two intact states (1 and 3), two repair states (2 and 4) and a defect state (5). Fig. IV.2 shows the CTMC Y_i as a labelled transition system, where the transition rates have been set to $1, 2, 3, 4$. Assume further that the overall system can only work if all of the m machines are in a local intact state. Since the set $A_i = \{1, 3\}$ contains the intact states of the i-th

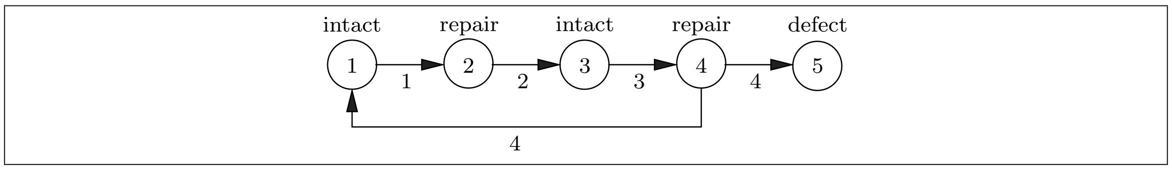


Figure IV.2: Example 1: Marginal CTMC Y_i, with initial state 1 and absorbing state 5.

machine, the set $A = \times_{i=1}^{m} A_i$ contains the global states, where all machines are in an intact state. Hence, $\mathbb{E}_A$ is the expected total time, where all m machines simultaneously are in an intact state. The mean time to absorption of Y would then, of course, be the mean time until all machines have reached their defect state.

Interpretation of example 2 Again, consider a system consisting of m parallel machines, where the i-th machine is modelled by the CTMC Y_i. Assume that the first four states $1, \ldots, 4$ of Y_i represent 4 different operational modes of the machine and state 5 represents failure of the machine. A machine can fail in each mode, but the rates of failure are different in each operational mode. Then $\mathbb{E}_A$ is the expected total time, where each of the m machines is in one of the two operational modes 1 or 3. The mean time to absorption of Y would then, of course, be the mean time until all machines fail.

IV.5.2 Three Iterative Methods

We computed the mean time to absorption $MTTA = \mathbb{E}[H]$ and the expected total time $\mathbb{E}_A$ using the exact and the approximate version of our novel approach which in both cases we refer to as Compositional Uniformisation (CU). Furthermore, we carried out computations with the approach based on Direct Uniformisation (DU) and the method of Jacobi Stepping (JS).

DU and JS were implemented in MATLAB 7.1 such that the optimised built-in functions of MATLAB for vector-matrix multiplications were employed. All matrices were implemented as sparse matrices. CU was implemented in JAVA.

Compositional And Direct Uniformisation For CU and DU we set out from the following sums in (IV.16)

$$MTTA = \frac{1}{q}\sum_{n=0}^{\infty}(1-\nu(n)(S)) \quad \text{and} \quad \mathbb{E}_A = \frac{1}{q}\sum_{n=0}^{\infty}\nu(n)(A),$$

and employ the CU method, DU respectively, to compute the values $\nu(n)(A)$ and $\nu(n)(S)$. Of course, to compute the above sums, a truncation index N must be introduced, such that we actually compute

$$MTTA(N) = \frac{1}{q}\sum_{n=0}^{N}(1-\nu(n)(S)) \quad \text{and} \quad \mathbb{E}_A(N) = \frac{1}{q}\sum_{n=0}^{N}\nu(n)(A),$$

For CU and DU the truncation index N was chosen as the smallest number where the relative change of two successive estimates for the $MTTA$ and $\mathbb{E}_A$ became smaller than $\epsilon = 10^{-6}$, i.e.,

$$\frac{MTTA(N) - MTTA(N-1)}{MTTA(N)} < \epsilon,$$
$$\frac{\mathbb{E}_A(N) - \mathbb{E}_A(N-1)}{\mathbb{E}_A(N)} < \epsilon.$$

For the approximate version of CU we chose the error parameter $\gamma = 1 - \sqrt[m-1]{1-10^6}$ (cf. remark 4), if dealing with m parallel Markov chains. That means, instead of the values $\nu(n)(\mathcal{A})$, $\mathcal{A} \in \{A, S\}$, which are needed to compute $\mathbb{E}_A$ and the $MTTA$, we computed approximations $\widetilde{\nu}(n)(\mathcal{A})$, with

$$0 \le \nu(n)(\mathcal{A}) - \widetilde{\nu}(n)(\mathcal{A}) \le 10^{-6}.$$

Jacobi Stepping In order to apply the Jacobi method, recall that according to (IV.2) and (IV.3) the $MTTA$ and $\mathbb{E}_A$ can be determined by the following. First solve

$$xT = \alpha, \tag{IV.25}$$

where T is the transient sub-matrix of Q and α the initial distribution of the transient states. Afterwards compute

$$MTTA = -x1 \quad \text{and} \quad \mathbb{E}_A = -x1_A,$$

where 1 is a column vector consisting of ones only, and 1_A is a column vector where entries corresponding to the set A are one, and all other entries are zero.

Of course, the critical part is the solution of (IV.25). To compute x the following iteration scheme (Jacobi Stepping) is used

$$x^{(k+1)} = (\alpha - x^{(k)}(T - T_D))T_D^{-1},$$

where T_D is a diagonal matrix containing the diagonal entries of T. The starting vector $x^{(0)}$ was chosen as a vector consisting of ones only. We stopped the iteration at the smallest index K, where both of the following criterions were satisfied:

$$\frac{|-x^{(K)}1 + x^{(K-1)}1|}{|x^{(K)}1|} < \epsilon, \quad \frac{|-x^{(K)}1_A + x^{(K-1)}1_A|}{|x^{(K)}1_A|} < \epsilon, \tag{IV.26}$$

i.e., the relative change of the estimates of the $MTTA$ and $\mathbb{E}_A$ obtained from two successive iteration vectors becomes smaller than $\epsilon = 10^{-6}$.

To check the quality of this criterion we conducted additional runs and determined K by evaluating the residual error, i.e., here K is the smallest index satisfying $\| \alpha - x^{(K)}T \|_\infty < \epsilon$. We found that in comparison to (IV.26) the number of iteration steps differed about at most ± 5. However, in order to minimise the processing time of the Jacobi method, for the results in the tables and graphs below the stopping criterion (IV.26) was used.

IV.5.3 Results

All of our computations were carried out on a system equipped with an Intel Pentium M 740 processor (1.73 GHz) and 1 GB main memory.

For the Jacobi method and the method of direct uniformisation computations were carried out for the values $m = 2 \dots 8$. All required matrices and vectors were kept in main memory using an explicit representation (as sparse structures). For larger values of m the required space exceeded the available main memory. We note that due to the representation of Q as the Kronecker sum $Q = \oplus_{i=1}^{m} Q_i$ it would be possible to access the entries of Q without explicitly storing the entire matrix, such that storage requirements are not really the bottleneck. However, such a procedure would not reduce the computation times of these methods. For the CU method m ranged from 2 to 20, and the memory consumption was negligible.

Table IV.1 and table IV.2 show the numerical results (truncated after the first 4 decimal positions) for the examples 1 and 2. CU refers to both the exact and the approximate version of compositional uniformisation. In the first example the results of the exact and the approximate CU had at least 6 matching post decimal positions, for all considered values of m and the given error parameter γ. In the second example we counted at least 7 matching post decimal positions. Furthermore, the truncation index N was the same for the exact and the approximate version of the CU method, for all considered m.

In Fig. IV.3 and Fig. IV.4 the computation times of the three methods CU, DU and JS are compared. Since the dimensions of the matrices P and Q which are being operated

Table IV.1: Numerical results for example 1.

m		2	4	6	8	10	20
number of states		25	625	$15,625$	$390,625$	$9,765,625$	$\approx 10^{13}$
JS (Jacobi Stepping)	$MTTA$	5.5407	7.4129	8.5882	9.4467	$*$	
	$\mathbb{E}_A$	1.1384	0.4473	0.2542	0.1718	$*$	
	steps K	100	149	196	243	$*$	
DU (Direct Uniformisation)	same results as CU below					$*$	
CU (Compositional Uniform.)	$MTTA$	5.5404	7.4121	8.5867	9.4446	10.1205	12.2599
	$\mathbb{E}_A$	1.1384	0.4473	0.2542	0.1718	0.1283	0.0559
	steps N	581	1133	1675	2213	2747	5386

Table IV.2: Numerical results for example 2.

m		2	4	6	8	10	20
number of states		25	625	$15,625$	$390,625$	$9,765,625$	$\approx 10^{13}$
JS (Jacobi Stepping)	$MTTA$	0.7370	1.0053	1.1730	1.2953	$*$	
	$\mathbb{E}_A$	0.1411	0.0608	0.0381	0.0276	$*$	
	steps K	93	134	174	212	$*$	
DU (Direct Uniformisation)	same results as CU below					$*$	
CU (Compositional Uniform.)	$MTTA$	0.7370	1.0052	1.1729	1.2952	1.3916	1.6967
	$\mathbb{E}_A$	0.1411	0.0608	0.0381	0.0276	0.0217	0.0104
	steps N	201	395	587	776	965	1897

on in the methods DU and JS in each iteration step grow exponentially in the degree m of parallelism, so do the computation times (note the log scaled Y-axis). In contrast to that, the computation times of the CU methods grow much slower according to the time complexity derived in (IV.17).

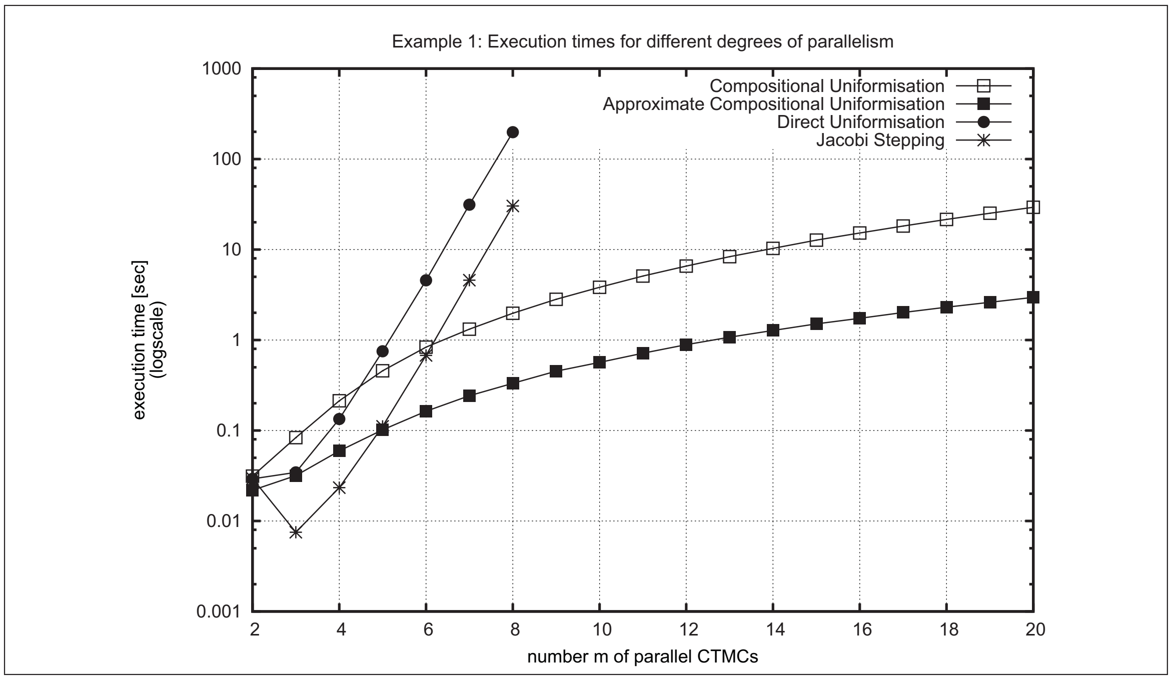


Figure IV.3: Example 1.

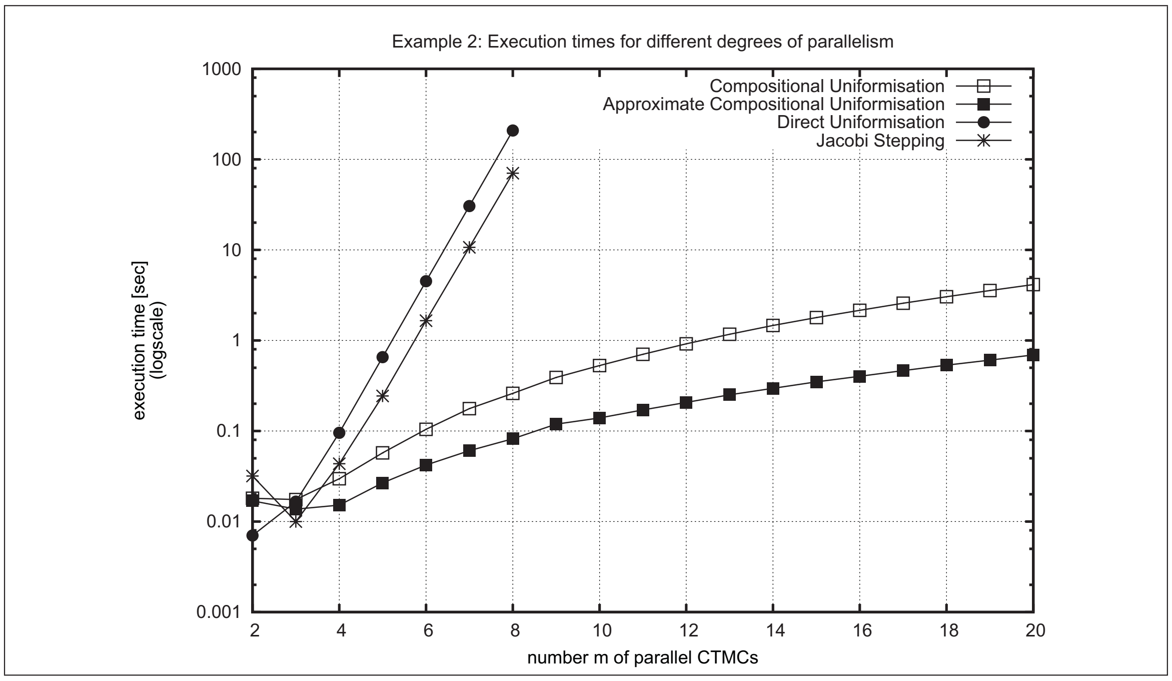


Figure IV.4: Example 2.

IV.6 CU: A Special Case with an Exact Solution

An interesting special case arises if in each Y_i all non-zero transition rates are equal, say q_i and there exist no cycles. Then Y_i is already uniformised, namely with rate q_i, and the number of steps until absorption is finite. That means, if N_i is the number of transient states of Y_i, then Y_i can perform at most N_i steps until absorption, or the other way round, Y_i has definitely become absorbed until the N_i-th step. Consequently, for each $i \in \{1, \dots, m\}$ we have $\nu_i(n)(S_i) = 1$, for $n \geq N_i$. This implies, that for every transient set $T_i \subset E_i$ of the marginal state space of Y_i

$$\nu_i(n)(T_i) = 0, \quad \text{for } n \geq N_i. \tag{IV.27}$$

Note, that here we use the Symbol T_i to represent a subset of E_i instead of the symbol A_i. We made this choice to stress the convention (earlier introduced in this paper) that the subsets A_i do not need to be transient, but the cross-product $\times_{i=1}^m A_i$ has to be a set of (global) transient states. Opposed to that, we require each T_i to be transient, which of course makes $\times_{i=1}^m T_i$ a transient set also.

From the definition of the $\star-$operator it is immediately seen that

$$\left(\star_{i=1}^m \nu_i[T_i]\right)(n) = 0, \quad \text{for } n \geq N := \sum_{i=1}^m N_i. \tag{IV.28}$$

According to (IV.16) and Theorem A (ii) the expected total time $\mathbb{E}_T$ which Y spends in the set $T := \times_{i=1}^m T_i$ is given by

$$\mathbb{E}_T = \frac{1}{q} \sum_{n=0}^{\infty} \left(\star_{i=1}^m \nu_i[T_i]\right)(n).$$

With (IV.28) and $N := \sum_{i=1}^m N_i$, we have

$$\mathbb{E}_T = \frac{1}{q} \sum_{n=0}^{N-1} \left(\star_{i=1}^m \nu_i[T_i]\right)(n).$$

Since this is a finite series, $\mathbb{E}_T$ can exactly be computed by employing Compositional Uniformisation.

As a concrete example, take $m = 10$ absorbing CTMCs Y_i, $i = 1 \dots m$, whose time to absorption is Erlang-distributed. Let Y_i consist of $5 + i$ states, where 1 is the initial state and $5 + i$ is the absorbing state. The transition rates shall be given by either $q_i = i$ or zero, as specified in Fig IV.5.

Let $T_i = \{1, \dots, 4 + i\}$, i.e., T_i contains all transient states of Y_i, and be $T := \times_{i=1}^m T_i$. Then, $\mathbb{E}_T$ is the expected total time, during which all of the marginal CTMCs are not

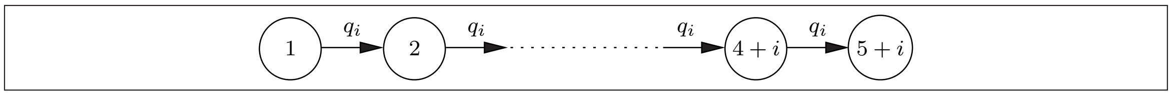


Figure IV.5: Marginal CTMC Y_i, with initial state 1 and absorbing state $5+i$.

Table IV.3: Numerical results for example 3.

m		2	4	6	8	10	20
CU (Compositional Uniformisation)	$\mathbb{E}_T$	2.7434	1.6045	1.2475	1.0794	0.9838	0.8125
	N	11	26	45	68	95	290

absorbed. In other words, this is the expected time instant, where the first (the fastest) marginal CTMC becomes absorbed. Still in other words, $\mathbb{E}_T$ is the expectation of the minimum of all marginal absorption times.

For a given number m of marginal CTMCs, the index N is given by $N := \sum_{i=5}^{m+4} i$. Table IV.3 shows the exact results for $\mathbb{E}_T$ as computed by the exact version of Compositional Uniformisation, for different degrees m of parallelism, as well as the associated index N. For the highest value $m = 20$, the processing time was about 200 milli-seconds. For $m = 20$, the joint CTMC Y possesses more than 10^{16} states. Hence, it should be clear that other methods which rely on the construction of the entire state space will consume prohibitively much time to compute the measure $\mathbb{E}_T$.

IV.7 Conclusion

In this chapter the novel method of *Compositional Uniformisation* for the computation of the mean time to absorption and the expected total time in some set of states of an absorbing continuous time joint Markov chain with independent marginal CTMCs was presented. In contrast to traditional techniques the generator of the joint Markov chain does not need to be constructed nor being operated on. Thus, the proposed algorithm is not subject to the state space explosion problem, but rather depends on the number of steps of the joint CTMC until absorption. It is based on at first processing the marginal CTMCs in isolation via a uniformisation procedure. Afterwards the marginal results are combined via a convolution-like operator. Numerical examples demonstrate a promising speed-up of the proposed algorithm in comparison with traditional techniques.

Chapter V

Extensions of the 2-LDS

In Chapter III we introduced the two-level decomposition scheme (2-LDS) which is applicable to PEPA models which possess an ergodic underlying Markov chain. In this chapter we aim at expanding the scope of the 2-LDS in two directions. Section V.1 deals with the computation of certain transient measures of terminating PEPA models in a compositional way. The resulting decomposition procedure is referred to as 2-LDS(T). In Section V.2 we consider MPA models in which the barrier-type synchronisation is replaced by a mechanism which we refer to as pre-emptive synchronisation. In a barrier synchronisation components which are ready to synchronise must wait for all other processes to become ready to synchronise on their part. Opposed to that, if the processes are involved in a pre-emptive synchronisation, the first process which becomes ready to synchronise initiates the synchronisation immediately. We give an informal description of an MPA which is able to model pre-emptive synchronisations, as well as a sketch of an algorithm which solves models generated by this MPA in a compositional fashion. This algorithm is referred to as 2-LDS(P).

V.1 Terminating PEPA Processes: 2-LDS(T)

The two-level decomposition scheme (2-LDS) introduced in Chapter III is applicable to PEPA models which possess an ergodic underlying Markov chain. Here, we extend the applicability to terminating PEPA models in the sense of [44], where the PEPA language is extended by the predefined component *stop*. Of course, instead of steady-state measures one can only deal with transient measures when dealing with terminating processes. We propose a solution technique for PEPA models with terminating behaviour in the style of the 2-LDS (in fact, most parts of the 2-LDS are reused). We refer to the resulting decomposition scheme as 2-LDS(T), where T stands for *terminating*.

V.1.1 Terminating Behaviour in PEPA

Termination of a PEPA component can be implemented as

(a) a deadlock,

(b) a component which repeats itself infinitely,

(c) explicit termination by including a *stop* component in the PEPA formalism.

The composite component C defined by

$$\begin{aligned} C &:= C_1 \underset{\{\alpha,\beta\}}{\bowtie} C_2, \\ C_1 &:= (\alpha, g).C_1, \\ C_2 &:= (\beta, h).C_2 \end{aligned}$$

possesses a deadlock. It does not execute any activities, and therefore it is a terminating component.

Let $Finish$ be a component which infinitely returns to itself after executing the activity $(null, a)$. Then a component C which eventually evolves into component $Finish$ can be interpreted as a terminating component:

$$\begin{aligned} C &:= (\alpha, g).Finish, \\ Finish &:= (null, a).Finish. \end{aligned}$$

In [44] an explicit null component *stop* is proposed which allows the modelling of components which cease to execute any activities. If e.g., $C = (\alpha, a).stop$, then C performs the activity (α, a) and afterwards terminates. The introduction of the null component *stop* requires a modification of the cooperation operator.

Definition 8. Let *stop* be a predefined component within the PEPA language which can not execute any activities. Cooperating behaviour involving the stop component is defined by

$$\begin{aligned} stop \underset{L}{\bowtie} stop &= stop, \\ C \underset{L}{\bowtie} stop &= C^{-L}, \end{aligned}$$

where C^{-L} is the component C with all actions $\alpha \in L$ blocked.

$\square$

If of two cooperating components one component stops, then the other component C proceeds to evolve as long as no synchronisation is required. If, however, C wants to synchronise, then the cooperation partner will never be ready for synchronisation (it has stopped), hence, C becomes blocked.

In the following, terminating behaviour of type (b) will not be considered because it can not be detected syntactically. A component which repeats itself infinitely can be viewed as a form of termination only if an additional layer of interpretation is imposed.

V.1.2 Sketch of the 2-LDS(T) for Terminating PEPA Processes

Here, we aim at transferring the ideas underlying the 2-LDS which deals with PEPA components in steady-state to the 2-LDS(T) which is to be applied to transient or absorbing models. Of course, this denies the possibility to compute steady-state probabilities of the considered model. Let the model under investigation be of the form

$$C = C_1 \underset{L}{\bowtie} C_2 \underset{L}{\bowtie} \cdots \underset{L}{\bowtie} C_m,$$

where the C_i are sequential components.

V.1.2.1 Target Quantities

We anticipate that, just like in the case of the 2-LDS, the application of the 2-LDS(T) to the terminating component C will yield a number of subsystems, where the dynamics between the subsystems is described by an embedded DTMC (first level of compositionality). An abstract sketch of such a model, where the decomposition into subsystems is already indicated, is given in Fig. V.1.

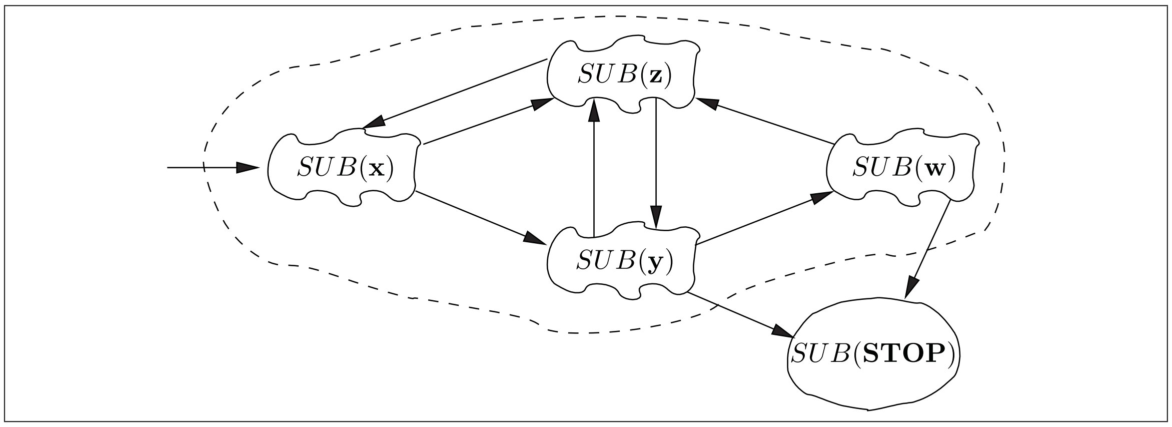


Figure V.1: Transient System.

Fig. V.1 indicates that there exists a subsystem $SUB(\mathbf{STOP})$. Assume that the only global terminating state is the state **STOP** and that $SUB(\mathbf{STOP})$ consists of that single state. Later on this will be justified by aggregating all global terminating states of the composite PEPA component into the single state **STOP**.

Since the model is transient, it cannot be solved for steady-state probabilities. But it is possible to compute expected total times which the entire system spends in a given subset A of the state space. Let A be a subset of the state space of C, where $A = \times_{i=1}^{m} A_i$ is the cross-product of subsets of the local state spaces, i.e., A_i is a subset of the state space of C_i, $i = 1, \ldots, m$. Furthermore, let A contain either only global synchronising states or only global non-synchronising states. In the case of non-terminating PEPA models these restrictions on the set A were required, in order to exploit the second level of compositionality, i.e., the compositional solution of the individual subsystems. For the same reason we adopt these restrictions to the current case of terminating PEPA models.

To solve the subsystem $SUB(\mathbf{x})$ means to compute the following two quantities:

$$\begin{aligned} T(\mathbf{x}), \mathbf{x} \in E^{(X)} &: \text{mean sojourn time in } SUB(\mathbf{x}) \\ T(\mathbf{x})(A), \mathbf{x} \in E^{(X)} &: \text{expected total time which } SUB(\mathbf{x}) \text{ spends in set } A. \end{aligned}$$

Note, that these two quantities relate to one cycle of the subsystem $SUB(\mathbf{x})$.

The following quantities give information about accumulated mean times before the entire system is exited. E.g., if $SUB(\mathbf{x})$ is visited 3 times on average before the entire system is exited, then $3 \cdot T(\mathbf{x})$ would be the accumulated mean sojourn time of $SUB(\mathbf{x})$. Let $\mathbb{V}_{\mathbf{x}}$, $\mathbf{x} \in E^{(X)}$, be the visit count of $\mathbf{x}$, i.e., the expected number of times the subsystem $SUB(\mathbf{x})$ is visited before the entire system is exited. Then the following quantities are easily understood:

$$\begin{aligned} \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x}) &: \text{accumulated mean sojourn time of } SUB(\mathbf{x}) \\ \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x})(A) &: \text{accumulated expected total time which } SUB(\mathbf{x}) \\ & \text{ spends in set } A \\ \sum_{\mathbf{x} \in E^{(X)} \setminus \{\mathbf{STOP}\}} \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x}) &: \text{mean sojourn time of the entire system} \\ \sum_{\mathbf{x} \in E^{(X)} \setminus \{\mathbf{STOP}\}} \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x})(A) &: \text{expected total time which the entire system spends in } A \end{aligned} \tag{V.1}$$

A way to compute the visit counts is given later in the context of the construction of the embedded DTMC.

V.1.2.2 The Requirements

Since we consider only terminating PEPA processes, the component C cannot be cyclic. Of course, this will deny the possibility to calculate steady-state probabilities of the underlying Markov chain of C. For the 2-LDS(T), we adopt the following two requirements from the 2-LDS for the same reasons as stated in Section III.4.

- All components synchronise over the same set L of actions.
- There exists no choice between synchronising and non-synchronising activities.

V.1.2.3 The Embedded DTMC

In order to construct the embedded DTMC X, the set $E^{(X)}$ of states which are embedded in C must chosen first. In the case of 2-LDS this set consisted of the global synchronising states (set S) and global states which can be reached immediately after a synchronising transition (set NS). If $\mathbf{x} = (x_1, \ldots, x_m) \in S$, then all x_i were local synchronising states. If $\mathbf{x} \in NS$, then at least one of the x_i was a local non-synchronising state, and all the other x_i were local synchronising states. If, in addition to that, we allow one or more of the entries of $\mathbf{x}$ to be a local *stop* state, then the set of embedded states can be divided into

The set S of global synchronising states: For $\mathbf{x} \in S$, all entries of $\mathbf{x}$ are local synchronising states.

The set NS of global non-synchronising states: For $\mathbf{x} \in NS$, at least one of the entries of $\mathbf{x}$ is a local non-synchronising state. All other entries are either local synchronising states or local *stop* states.

The global stop state STOP: Let $\mathbf{x}$ be a global state. If all entries are local *stop* states, it is clear that the $\mathbf{x}$ is a global *stop* state. If at least one entry is a local *stop* state and all other entries are local synchronising states, then $\mathbf{x}$ must also be a global terminating state because the synchronisation is blocked forever by the local stop-states. That means, if at least one entry of $\mathbf{x}$ is a local *stop* state and all other entries are local synchronising states, then $\mathbf{x}$ is a global terminating state. Assume that all global terminating states are aggregated into a single global state. We refer to this aggregated state as **STOP**.

The global stop state START: In the case of the 2-LDS, where ergodicity of the Markov chain underlying the composite PEPA component is assumed, the initial state of the composite component is only of importance in so far as it determines the irreducibility class of the global embedded states, i.e., only global embedded states which are reachable from the initial state constitute the set $E^{(X)}$. Furthermore, since the steady-state probabilities of the composite component do not depend on the initial state, the initial state could be assumed to be any of the states contained in S or NS. In the case of terminating PEPA processes, where only transient measures can be calculated, the behaviour of the system depends on the initial state. That means, in the case that the initial state is neither contained in S nor in NS, it must be treated separately. Denote the initial state by **START**. If $\mathbf{START} \notin S$, $\mathbf{START} \notin NS$ and $\mathbf{START} \neq \mathbf{STOP}$, then **START** must be added to the set $E^{(X)}$.

For the 2-LDS(T) the set of embedded states of a terminating PEPA component C is given by

$$E^{(X)} = S \cup NS \cup \{\mathbf{START}, \mathbf{STOP}\}.$$

The algorithm to determine the set $E^{(X)}$ as well as the embedded DTMC could work along the lines of the algorithm given for the 2-LDS in Section III.5. This algorithm explored the local synchronising states and local states which can be reached immediately after a synchronisation. Out of these local states the global embedded states were constructed. The transition probabilities of the embedded DTMC X were also calculated during this exploration scheme. In the current case of the 2-LDS(T) this algorithm must be extended such that local stop states are incorporated into the exploration scheme of local states. The explicit listing of such an algorithm should not present a severe problem and is left as an exercise for the interested reader.

V.1.2.4 Visit Counts of Subsystems

Let X be the embedded DTMC which describes the behaviour between the subsystems. Since $SUB(\mathbf{STOP})$ is the only absorbing subsystem, with the single absorbing state **STOP**, X is an absorbing DTMC with the absorbing state **STOP**.

Let X be a finite absorbing DTMC, with state space $E^{(X)} \cup \{\mathbf{STOP}\}$, where **STOP** is the only absorbing state of X. Let G be the matrix which contains only the one-step transition probabilities between transient states, i.e., in order to obtain G, all rows and columns associated with absorbing states are removed from the entire transition probability matrix $(\mathbb{P}(X_1 = j | X_0 = i))_{i,j \in E^{(X)}}$. Let $V = (v_{i,j})_{i,j \in E^{(X)} \setminus \{\mathbf{STOP}\}}$ be the matrix of visit counts, i.e., $v_{i,j}$ is the expected number of visits to state j, provided that $X_0 = i$. If $i = j$, then the initial value $X_0 = i = j$ is counted as one visit. Then, V is given by

$$V = (I - G)^{-1}. \tag{V.2}$$

Let α be the initial probability distribution of transient states and let $\mathbb{V}_j$, $j \in E^{(X)}$, be the expected number of visits to the state j, given the initial distribution α of transient states of X. Then, for $j \in E^{(X)} \setminus \{\mathbf{STOP}\}$,

$$\mathbb{V}_j = \alpha \cdot (I - G)^{-1} \cdot 1_j, \tag{V.3}$$

where 1_j is a column vector where the entry corresponding to state j is 1, and all other entries are zero.

V.1.2.5 The Subsystems

A subsystem $SUB(\mathbf{x})$, $\mathbf{x} \in E^{(X)}$ is defined by its entry state $\mathbf{x}$. In the 2-LDS we had two types of subsystems. The subsystem $SUB(\mathbf{x})$, for $\mathbf{x} \in S$, consisted of only the state $\mathbf{x}$. The subsystem $SUB(\mathbf{x})$, for $\mathbf{x} \in NS$, could be represented by the parallel evolution of m independent absorbing CTMCs which were obtained by manipulation of the generator matrices underlying the m sequential PEPA components $C_1, \ldots, C_m$.

In the 2-LDS(T) we have the following four types of subsystems.

$SUB(\mathbf{x})$, $\mathbf{x} \in S$: Obviously, this subsystem consists of the single state $\mathbf{x}$. It is solved exactly as in the case of non-terminating PEPA components.

$SUB(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_m) \in NS$: In the subsystem $SUB(\mathbf{x})$ the m components start to evolve independently of each other in the local states $x_1, \ldots, x_m$. The subsystem $SUB(\mathbf{x})$ is left when each of the m components has either reached a local synchronising state or a local stop state. If in the m components all local synchronising states and stop states are declared as absorbing, then the time instant where the slowest of the components becomes absorbed corresponds to the time instant where the subsystem $SUB(\mathbf{x})$ is left. For $i = 1, \ldots, m$, let Y_i be the CTMC which results from component C_i by declaring all (local) synchronising states and stop states as absorbing. Let the state x_i be the initial state of Y_i. Then, the absorbing joint CTMC $Y := (Y_1, \ldots, Y_m)$, with initial state $\mathbf{x} = (x_1, \ldots, x_m)$, describes the behaviour of the subsystem $SUB(\mathbf{x})$. Y can be solved with the methods discussed in Chapter IV. Note, that the construction of $Y = (Y_1, \ldots, Y_m)$ is almost identical to the construction of the absorbing CTMC in the 2-LDS in section III.5. The only difference is that, here, during the construction of the Y_i's not only the local synchronising states but also the local *stop* states of the sequential PEPA components are declared as absorbing.

$SUB(\mathbf{START})$: If $\mathbf{START} \notin S \cup NS \cup \{\mathbf{STOP}\}$ then it can be solved in the same way as if it were contained in NS.

$SUB(\mathbf{STOP})$: This subsystem consists of the single absorbing state **STOP**. It doesn't need to be solved.

V.1.3 Example: A simple Producer-Consumer System with Failures

Consider the following simple producer-consumer scenario given as a PEPA model.

$$C = Producer \underset{\{handover\}}{\bowtie} Consumer$$
$$Producer = [(produce, p) + (fail, f).stop].(handover, h).Producer$$
$$Consumer = (handover, h).[(consume, c).Consumer + (fail, f).stop]$$

The component *Producer* either produces one item or fails. In the latter case the component is stopped. If an item is successfully produced it is handed over to the consumer. After that, the producer enters a new production cycle. Once the component *Consumer* has received the item, it can either consume it or fail.

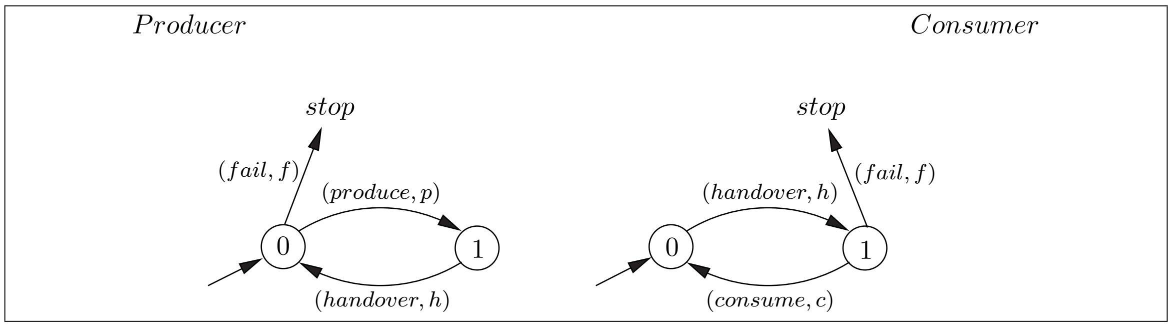


Figure V.2: Producer-Consumer system.

It is easily seen that sooner or later this system will fail. Eventually one of the components will reach its local state *stop*. Once this has happened, the other component will eventually reach either the local state *stop*, in which case the composite system stops by definition, or a local synchronising state, in which case that component becomes blocked. In both cases the system terminates; we interpret this termination as a system failure.

Our exemplary measures of interest are the mean time to failure (MTTF) and the mean time, where the producer produces an item and simultaneously the consumer consumes an item, i.e., the mean time the system spends in the state $(0, 1)$.

V.1.3.1 The Embedded DTMC

For this simple system, the following global embedded states can easily be determined by looking at Fig. V.2. Of course, the initial state is given by $\mathbf{START} = (0, 0)$. We will treat this state later. The only synchronising state is the state $(1, 0)$. It leads with probability one to the embedded state $(0, 1)$ which is the only state which can be reached immediately after a synchronisation. That means up to now we have: $S = \{(1, 0)\}$, $NS = (0, 1)$, $\mathbb{P}((\mathbf{1}, \mathbf{0}) \rightarrow (\mathbf{0}, \mathbf{1})) = 1$.

Embedded successor states of $(0, 1) \in NS$ We briefly recall the general procedure in the 2-LDS. Let $\mathbf{x} = (x_1, \ldots, x_m) \in NS$. In the case of non-terminating PEPA components the procedure was to declare local synchronising states as absorbing in all local components. Then, absorption probabilities of the local components were computed, where the x_i were taken as the initial states of the components. Global (or joint) absorbing states were the embedded successor states of $\mathbf{x}$ and the absorption probabilities of the global absorbing states were the transition probabilities to the embedded successor states. Absorption probabilities of global absorbing states were obtained by multiplication of local absorption probabilities. In the case of terminating PEPA components, i.e., in the case of the 2-LDS(T), the procedure is almost identical, except that we also have to deal with local stop states. Since a (local) component ceases to execute any activities, once it has reached a stop state, i.e., it remains in the stop state forever, it is clear that local stop states have to be declared as absorbing. For the concrete state $(0, 1) \in NS$ of the producer-consumer model this is illustrated in Fig. V.3.

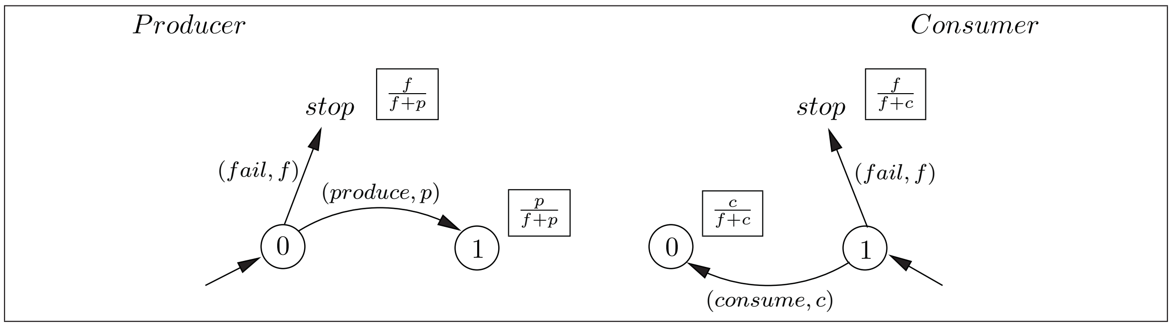


Figure V.3: Modified Producer-Consumer system. Initially the components are in the local states 0, 1 respectively. Local synchronising states are declared as absorbing and local stop states are absorbing anyway. Local absorption probabilities are surrounded by rectangles.

From Fig. V.3 it is seen that the state $(0, 1)$ possesses the following four embedded successor states as well as the given transition probabilities:

$(stop, 0)$: This is a global terminating state, since the component $Consumer$ is blocked in the state 0. $\mathbb{P}((0, 1) \to (stop, 0)) = \frac{f \cdot c}{(f+p)(f+c)}$.

$(1, stop)$: This is a global terminating state, since the component $Producer$ is blocked in the state 1. $\mathbb{P}((0, 1) \to (stop, 0)) = \frac{p \cdot f}{(f+p)(f+c)}$.

$(stop, stop)$: This is a global *stop* state by definition. $\mathbb{P}((0, 1) \to (stop, 0)) = \frac{f \cdot f}{(f+p)(f+c)}$.

$(1, 0)$: This state was already identified as the only synchronising state. $\mathbb{P}((0, 1) \to (1, 0)) = \frac{p \cdot c}{(f+p)(f+c)}$.

Of course, the three global terminating states $(stop, 0)$, $(1, stop)$ and $(stop, stop)$ are aggregated into the single state **STOP**.

Embedded successor states of START Since $\mathbf{START} = (0,0) \notin S \cup NS \cup \{\mathbf{STOP}\}$ it must be treated separately. It should be clear, that in this case the embedded successor states of **START** as well as the one-step transition probabilities to these successor states are obtained in the same way as if **START** were contained in NS. That means, in the local PEPA components are modified to yield absorbing CTMCs. The entries of the state $\mathbf{START} = (0,0)$ represent local starting states for the modified (absorbing) PEPA components. Local absorbing states, absorption probabilities respectively, are compined to yield global embedded successor states, one-step transition probabilities to the successor states respectively. The modification of the local components as well as the local absorption probabilities are illustrated in Fig. V.4.

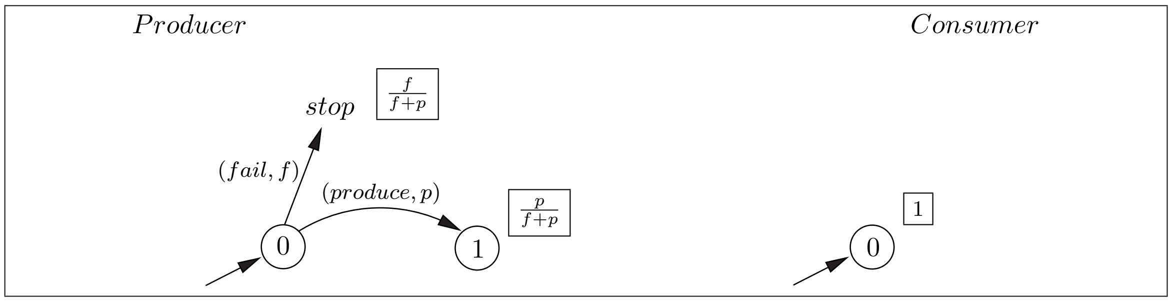


Figure V.4: Producer-Consumer system. Initially the components are in the local states 0, 0 respectively. Local synchronising states are declared as absorbing and local stop states are absorbing anyway. Local absorption probabilities are surrounded by rectangles.

From Fig. V.4 the following embedded successor states of **START** as well as the corresponding transition probabilities can be identified.

$(stop, 0)$: This global terminating state has already been identified. $\mathbb{P}((0,0) \rightarrow (STOP, 0)) = \frac{f}{f+p}$.

$(1,0)$: This state was already identified as the only synchronising state. $\mathbb{P}((0,0) \rightarrow (1,0)) = \frac{p}{(f)(f+p)}$.

We obtain the sets of embedded states

$$S = \{(1,0)\}, \qquad NS = \{(0,1)\}, \qquad \{\mathbf{START}, \mathbf{STOP}\}$$
$$\text{and } E^{(X)} = S \cup NS \cup \{\mathbf{START}, \mathbf{STOP}\} = \{(1,0), (0,1), \mathbf{START}, \mathbf{STOP}\}$$

and the transition probabilities of the embedded DTMC

$$\begin{aligned}
\mathbb{P}(\mathbf{START} \to \mathbf{STOP}) &= \frac{f}{f+p} \\
\mathbb{P}(\mathbf{START} \to (1,0)) &= \frac{p}{f+p} \\
\mathbb{P}((1,0) \to (0,1)) &= 1 \\
\mathbb{P}((0,1) \to \mathbf{STOP}) &= \frac{f \cdot (c+p+f)}{(f+p)(f+c)} \\
\mathbb{P}((0,1) \to (1,0)) &= \frac{p \cdot c}{(f+p)(f+c)}
\end{aligned}$$

V.1.3.2 Visit Counts

The transition probabilities between embedded states describe the dynamics between the subsystems as illustrated in Fig. V.5. The transition probability matrix G which captures

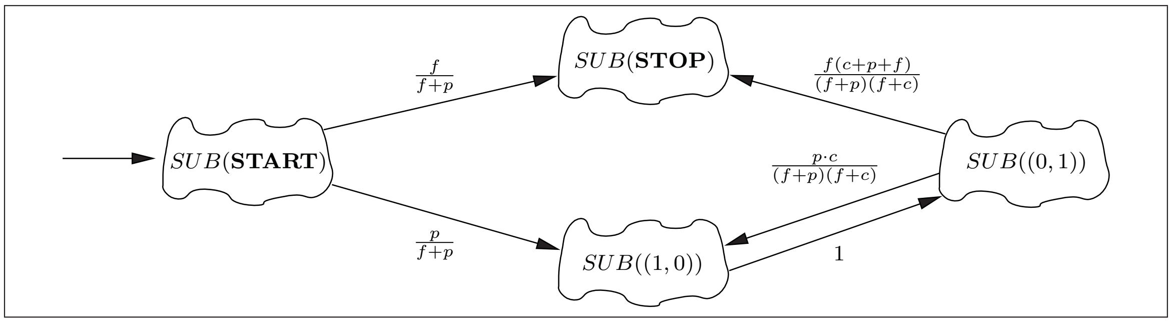


Figure V.5: Dynamics between subsystems of the Producer-Consumer system.

only the transitions between transient states and the matrix V of visit counts which results from (V.2) are given by

$$G = \begin{pmatrix} 0 & \frac{p}{f+p} & 0 \\ 0 & 0 & 1 \\ 0 & \frac{p \cdot c}{(f+p)(f+c)} & 0 \end{pmatrix}, \qquad V = \begin{pmatrix} 1 & \frac{p(f+c)}{f(f+c+p)} & \frac{p(f+c)}{f(f+c+p)} \\ 0 & \frac{(f+p)(f+c)}{f(f+c+p)} & \frac{(f+p)(f+c)}{f(f+c+p)} \\ 0 & \frac{p \cdot c}{f(f+c+p)} & \frac{p \cdot c}{f(f+c+p)} \end{pmatrix}.$$

According to (V.3), this yields the visit counts

$$\begin{aligned}
\mathbb{V}_{\mathbf{START}} &= 1, \\
\mathbb{V}_{(0,1)} = \mathbb{V}_{(1,0)} &= \frac{p(f+c)}{f(f+c+p)}.
\end{aligned}$$

V.1.3.3 The Subsystems and Solution of the Composite System

Recall that our measures of interest are the mean time to failure (MTTF) and the mean time the system spends in the state $(0,1)$ (MT(0,1)). According to (V.1) these measures are given by

$$\begin{aligned} MTTF &= \sum_{\mathbf{x}\in E^{(X)}\setminus\{\mathbf{STOP}\}} \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x}) \\ MT(0,1) &= \sum_{\mathbf{x}\in E^{(X)}\setminus\{\mathbf{STOP}\}} \mathbb{V}_{\mathbf{x}} \cdot T(\mathbf{x})(A), \end{aligned} \tag{V.4}$$

where $A = \{(0,1)\}$.

Since we already calculated the visit counts $\mathbb{V}_{\mathbf{x}}$, $\mathbf{x} \in E^{(X)}$, it remains to compute the mean sojourn time $T(\mathbf{x})$ of $SUB(\mathbf{x})$ and the expected total time $T(\mathbf{x})(A)$ which subsystem $SUB(\mathbf{x})$ spends in the set $A = \{(0,1)\}$, for all $\mathbf{x} \in E^{(X)}$.

Subsystem $SUB((1,0))$

Since $(1,0) \in S$, this subsystem consists of the single synchronising state $(1,0)$. This state possesses a single outgoing transition with rate h. Hence,

$$\begin{aligned} T((1,0)) &= \frac{1}{h}, \\ T((1,0))(A) &= 0. \end{aligned}$$

Subsystem $SUB(\mathbf{START})$

Since $\mathbf{START} = (0,0)$ is neither a terminating state nor a synchronising state, $SUB(\mathbf{START})$ is solved the same way as if $\mathbf{START}$ were contained in NS. Let Y_{prod} be the CTMC which results from the component *Producer* by declaring all local synchronising states as absorbing. Analogously, let Y_{cons} be the CTMC which results from the component *Consumer* by declaring all local synchronising states as absorbing. That means, Y_{prod} and Y_{cons} are the same absorbing CTMCs as those depicted in Fig. V.4.

Let $Y := (Y_{prod}, Y_{cons})$ be the absorbing joint CTMC, with initial state $\mathbf{START} = (0,0)$. Then $T(\mathbf{START})$ is the mean time to absorption of Y and $T(\mathbf{START})(A)$ is the expected total time which Y spends in the set $A = \{(0,1)\}$. Of course, for concrete values of the transition rates within the marginal CTMCs, these two quantities can be computed numerically by employing one of the methods of Chapter IV. E.g., we could use Compositional Uniformisation which computes the quantities in a compositional way from the marginal CTMCs. In our little example, however, the explicit construction of the joint CTMC Y

does not present a problem and immediately yields the following symbolic results:

$$T(\mathbf{START}) = \frac{1}{f+p},$$
$$T(\mathbf{START})(A) = 0.$$

Subsystem $SUB((0,1))$

Since $(0,1) \in NS$ the calculation of the quantities $T((0,1))$ and $T((0,1))(A)$ proceeds in complete analogy to the solution of $SUB(\mathbf{START})$. Let $(0,1)$ be the initial state of $Y = (Y_{prod}, Y_{cons})$. Then the marginal absorbing CTMCS Y_{prod} and Y_{cons} are the same as those depicted in Fig. V.3. Of course, $T((0,1))$ is the mean time to absorption of Y and $T((0,1))(A)$ is the expected total time which Y spends in the set $A = \{(0,1)\}$. The following results are obtained

$$T((0,1)) = \frac{1}{2f+c+p}\left(1 + \frac{f+p}{c+f} + \frac{c+f}{f+p}\right),$$
$$T((0,1))(A) = \frac{1}{2f+c+p}.$$

Solution of the Composite System

For the sake of completeness, we give the (symbolic) solution of the mean time to failure (MTTF) and the mean time the system spends in the set $A = \{(0,1)\}$. Substitution of the previously obtained results into (V.4) yields

$$MTTF = \underbrace{1}_{\mathbb{V}_{\mathbf{START}}} \cdot \underbrace{\frac{1}{f+p}}_{T(\mathbf{START})} + \underbrace{\frac{p(f+c)}{f(f+c+p)}}_{\mathbb{V}_{(0,1)}=\mathbb{V}_{(1,0)}} \left[\underbrace{\frac{1}{h}}_{T((1,0))} + \underbrace{\frac{1}{2f+c+p}\left(1 + \frac{f+p}{c+f} + \frac{c+f}{f+p}\right)}_{T((0,1))} \right],$$
$$MT(0,1) = \underbrace{\frac{p(f+c)}{f(f+c+p)}}_{\mathbb{V}_{(0,1)}} \underbrace{\frac{1}{2f+c+p}}_{T((0,1))(A)}.$$

V.2 Beyond Barrier Synchronisation: 2-LDS(P)

In the models which we investigated so far (2-LDS and 2-LDS(T)) the second level of compositionality resulted in subsystems which consist of either a single state (these subsystem contain global synchronising states, or the global **STOP** state) or can be represented by the parallel evolution of independent absorbing Markov chains, where time to absorption of the joint Markov chain coincides with the sojourn time of the subsystem. This was of consequence of the barrier-type synchronisation of the considered models, where all components which are willing to synchronise must wait for all other components to become ready to synchronise on their part.

In this section we consider a class of models with a different synchronisation mechanism which we refer to as **pre-emptive synchronisation**. A component which is ready to synchronise interrupts the other components and initiates the synchronisation immediately. This results in subsystems which still can be represented as the parallel evolution of absorbing Markov chains, but now the time until the first (i.e., the fastest) of the Markov chains becomes absorbed corresponds to the sojourn time of the subsystem. The decomposition scheme to be applied to these models will be referred to as 2-LDS(P), where P stands for *pre-emptive*.

Instead of an in-depth analysis of models with synchronisations, we explain our ideas by means of an example sensor network which consists of m clusters of sensors. In section V.2.1 a simple version of the sensor network with network failure, but without any synchronisation is described and analysed. In section V.2.2 the network is extended by a repair mechanism which is executed after network failure, where the repair can be seen as a pre-emptive synchronisation of the clusters of the network. A solution of the entire system in the style of the two-level decomposition scheme is sketched. The decomposition of the entire system into subsystems, though, is not necessary, since we explicitely construct the entire system out of given subsystems. In section V.2.3 we try to generalise our ideas to the case, where the subsystems are not known in advance. We informally define a Markovian Process Algebra which allows the implementation of pre-emptive synchronisations. Guidelines for the developement of a 2-LDS(P) for models which are generated by this MPA are given.

V.2.1 Example: Sensor Network without Repair

Consider a network which consists of m clusters of sensors. The clusters $1 \dots m$ each possess 100 sensors, where each sensor measures some cluster-specific data or executes some other cluster-specific task. The life times of the sensors of cluster i are exponentially distributed with rate d_i, e.g., caused by battery decay or environmental influences. A cluster is said to be available (or intact) if there are at least 20 sensors active per cluster. The entire network functions properly as long as all m clusters are intact. That means each of the clusters $1 \dots m$ must contain at least 20 active sensors. In reliability modelling it is common to represent such a system as a series configuration or series system as in the reliability block diagram in Fig. V.6. The meaning of such a configuration is that the entire system is available (intact) as long as there exists a path through the configuration. E.g., if cluster 2 is defective (i.e., less then 20 active sensors), then a path through the series system is not possible.

Each cluster of sensors represents a k-of-n system, where $k = 20$ and $n = 100$, i.e., in each cluster there must be at least k of the n sensors intact for the entire cluster to be available

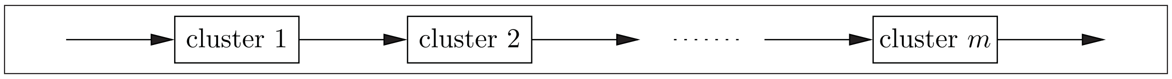


Figure V.6: A reliability block diagram of a series configuration consisting of the m units cluster $1, \ldots,$ cluster m.

(or intact). Since it is assumed that each sensor of cluster i possesses an exponentially distributed lifetime, with rate d_i, cluster i can be modelled as a pure Markovian death process, with death rate d_i, and the states $19, 20, 21, \ldots, 100$, where the state number indicates the number of active sensors. The absorbing state 19 represents the case where less than 20 sensors of the considered cluster are active, i.e., the cluster is not available. An illustration of the death process for cluster i is given in Fig. V.7. Of course, every unit in the series configuration in Fig V.6 is represented as one such Markovian death process. The lifetime of the series system is characterised by the parallel evolution of the m death processes. If the first death process becomes absorbed (the cluster becomes unavailable) the entire series system becomes unavailable.

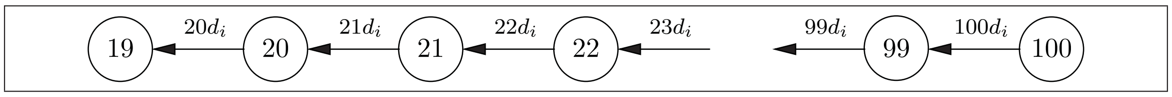


Figure V.7: A Markovian death process modelling cluster i. State numbers coincide with numbers of active sensors. Cluster i is considered unavailable if the death process is in state 19.

Reliability/Performance Measures The obvious reliability measure of interest is certainly the mean time to failure (MTTF) of the series system, i.e., the mean time until the first of the m clusters becomes unavailable. Since every cluster is modelled as a Markovian death process, where the absorbing state 19 corresponds to unavailability of the cluster, the entire system becomes unavailable as soon as the first of the death processes enters its absorbing state. Or, the other way round, the entire system is available as long as each of the death processes is in a local transient state. Let A_i be the set of all transient states of the i-th death process, i.e., $A_i = \{20, 21, \ldots, 100\}$. Let $A := \times_{i=1}^{m} A_i$. Then, the expected total time $\mathbb{E}_A$ which the series system spends in A equals the mean time to failure.

Assume, that the quality of the data gathered by the sensors depends on the number of intact sensors. If, the number of intact sensors of each cluster is above L, $20 \leq L \leq 100$, the gathered data is considered to be of a certain level of quality. For example, for $L = 90$, i.e., at least 90 sensors are intact in each cluster, the quality could by considered excellent, whereas otherwise it is considered only moderate or bad. We are interested in the mean time that the sensor network provides data of a certain level of quality, i.e., the mean time until the number of sensors in one cluster drops below L, or, equivalently, the mean time which all of the m death processes spend in a local state $\geq L$. This measure is commonly referred to as the *uptime* of service level L. For $A_i = \{L, L+1, \ldots, 100\}$ and $A := \times_{i=1}^{m} A_i$

this is the expected total time $\mathbb{E}_A$ which the system spends in the set A. Note, that for $L = 20$ the uptime of the quality level $L = 20$ coincides with the uptime of the entire network, i.e., the mean time to failure of the network.

Fig. V.8 shows a numerical evaluation of the mean uptime and the standard deviation as functions of the quality level L. The computations were carried out with the method of *Compositional Uniformisation*, where the death rates of the Markovian death processes were set to $d_i = i$, for $i = 1, \ldots, m$, and $m = 10$.

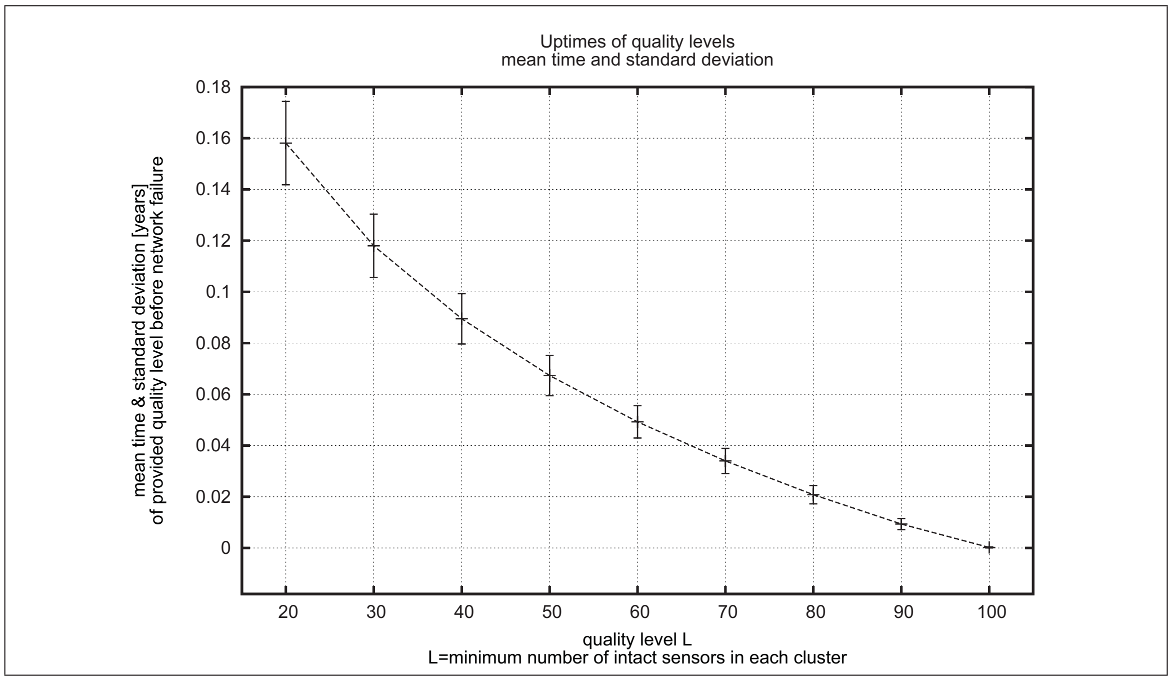


Figure V.8: Uptimes of quality levels of the sensor network.

V.2.2 Example: Extended Sensor Network with Repair

We extend this scenario by that if the network breaks down, i.e., if at least one of the clusters becomes defective, a repair process is initiated to fully repair the entire network which involves reparation or replacement of every defective sensor. This will introduce a pre-emptive synchronisation over all m clusters of sensors.

Intuitively we would model each of the m clusters as an MPA component, where the initial states would represent fully intact clusters (no defective sensors). Global synchronisation would then take place if one of the m clusters became defective. The synchronisation (repair process) would then transit the entire model into the initial state. Unfortunately, this scenario does not transform well into given MPA languages. In common MPA languages synchronisation is implemented as barrier synchronisation. Each component willing to synchronise must wait until all other components to be involved in the synchronisation become ready to synchronise on their part. But in our example a component which is ready to synchronise, i.e., a defective cluster of sensors, immediately initiates the synchronisation.

Instead of giving an MPA model of the overall system, and afterwards applying some sort of decomposition scheme on the model, here, we follow a slightly different path. First, we define the subsystems. After that, we construct the overall system out of the subsystems and the dynamics between the subsystems. We anticipate that due to the different preemptive synchronisation mechanism the structure of the subsystems will be a bit different to what we are used to from common MPA models. We will show that a slightly modified version of our two-level decomposition scheme is still applicable to the overall system.

We define two subsystems $SUB(\mathbf{z})$ and $SUB(\mathbf{s})$. $SUB(\mathbf{z})$ describes the behaviour of the system from the initial state $\mathbf{z} = (100, 100, \ldots, 100)$, where all sensors in all clusters are intact, up to the time instant where the first cluster becomes unavailable. Obviously, $SUB(\mathbf{z})$ is just the series system from the previous section. $SUB(\mathbf{s})$ describes the activity of the expedition team. If $SUB(\mathbf{z})$ is left, i.e., one cluster of sensors becomes unavailable, the overall system enters $SUB(\mathbf{s})$ (repair by expedition team). Upon leaving $SUB(\mathbf{s})$ (repair is finished) the overall system moves over to $SUB(\mathbf{z})$, i.e., a new cycle is started, where initially all sensors are active.

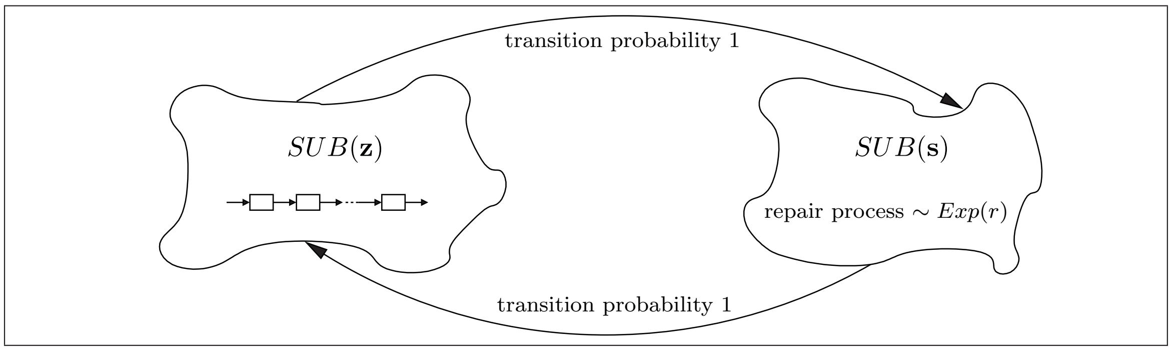


Figure V.9: Sensor network with repair.

Reliability/Performance Measures We solve the sensor network including the repair mechanism for an exemplary performance measure, where we use the same values for the parameters d_i and m as in the previous section, i.e., $d_i = i$, for $i = 1, \ldots, m$, and $m = 10$. Our target quantity is the availability of a quality level L, i.e., the steady-state probability that in each of the clusters at least L sensors are active. Let A be the set of all global states which indicate that at least L sensors of each cluster are active.

Then, in order to obtain the steady-state probability $\mathbb{P}(System \in A)$ with the two-level decomposition scheme it is required to

- establish the embedded DTMC and compute its steady-state distribution

- compute the mean sojourn times $T(\cdot)$ of the subsystems as well as the expected total times $T(\cdot)(A)$ which the subsystems spend in the set A.

The embedded DTMC alternates between the two states $\mathbf{z}$ and $\mathbf{s}$, and the steady-state probabilities are given by $\pi(\mathbf{z}) = \pi(\mathbf{s}) = 0.5$.

$SUB(\mathbf{s})$ consists of only one state. Its holding time is exponentially distributed with rate r. The mean sojourn time is then, of course, given by $T(\mathbf{s}) = \frac{1}{r}$. If the system is in this subsystem, there must be at least one cluster of sensors which is not available, hence, $T(\mathbf{s})(A) = 0$.

$SUB(\mathbf{z})$ is the series system from the previous section. It was shown how to compute the mean time to failure of the series system (equals mean sojourn time $T(\mathbf{z})$ of $SUB(\mathbf{z})$) and the expected total time in the subset A (equals $T(\mathbf{z})(A)$ of $SUB(\mathbf{z})$) in a compositional way, hence, we can adopt the results from the previous section. The only difference between this subsystem and subsystems obtained from common MPA models (with barrier synchronisations) is that the sojourn time of $SUB(\mathbf{z})$ is given by the time until the first of its constituent (marginal) Markov processes becomes absorbed as opposed to the common case, where the sojourn time is determined by the slowest of the constituent processes.

Fig. V.10 shows the availability of different quality levels. The results were obtained by substituting the gathered data ($\pi(\cdot)$, $T(\cdot)$, $T(\cdot)(A)$) into formula (III.3).

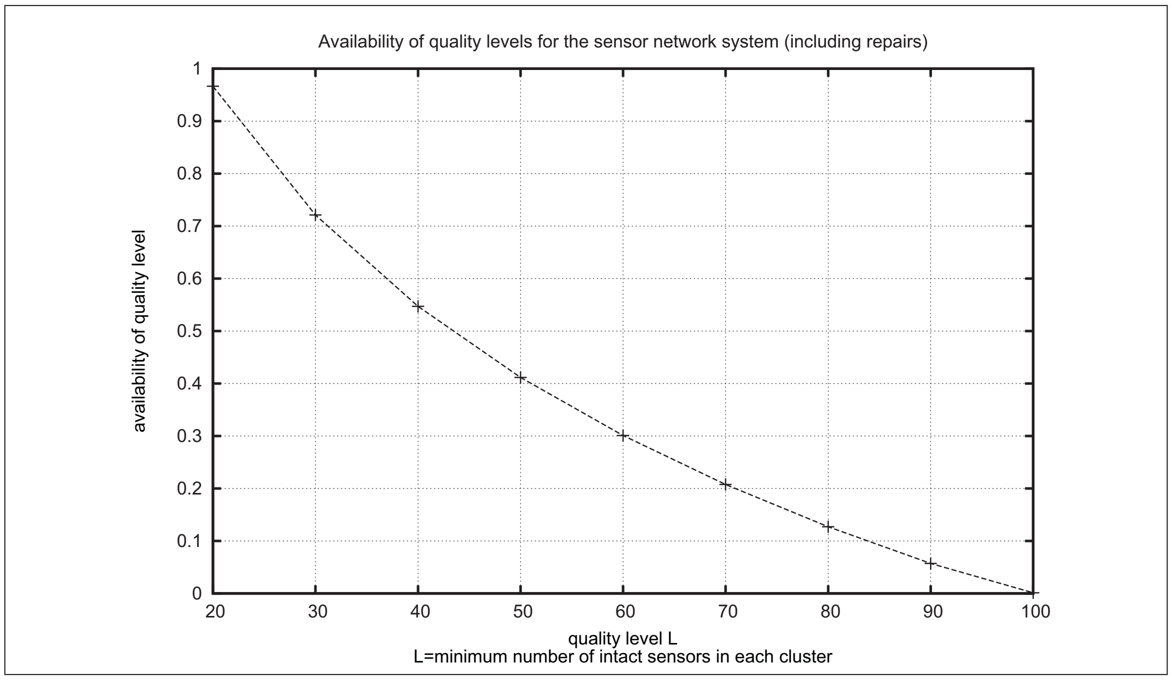


Figure V.10: Availability of quality levels of the sensor network with repair.

V.2.3 Sketch of the 2-LDS(P)-Algorithm

In this section we give hints at how the two-level decomposition scheme can be applied in the context of an MPA language which allows the modelling of pre-emptive synchronisation. Remember, in the preceding section we constructed an overall system out of subsystems of which we knew how to solve them. Here, we start from a given MPA model which includes the altered synchronisation and try to decompose it into subsystems which can be handled by the 2-LDS(P). Again, we illustrate the ideas by means of the sensor network example.

Our modification of the PEPA language consists of the replacement of the cooperation mechanism, in the sense that in the modified version it is possible that a component which is ready to synchronise can force other components to take part in the synchronisation immediately.

Assume that α is a synchronising action type in a PEPA model in the common sense. Then, components in which activities labelled with α are enabled must wait until all other synchronising components also have enabled an activity labelled with action type α. Synchronisation proceeds afterwards. We modify this synchronisation mechanism by introducing two new action types α^+ and α^-. Initially all α^--activities are switched off, i.e., they are ignored. A component in which an activity of type α^+ is enabled signals its willingness to synchronise. As a consequence the α^--activities in all components are switched on. Components with enabled α^--activities must wait until synchronisation occurs, even if also non-synchronising activities are enabled (i.e., no synchronisation timeout). Synchronisation proceeds if all of the other components to take part in the synchronisation have enabled an α^--activity, i.e., as soon as they have reached a local state with an outgoing α^--activity. The rate of synchronisation is determined by the rate of the α^+-activity. The striking difference between this synchronisation mechanism and common synchronisation is that fact, that the α^--activities do not influence the behaviour of a component unless another component currently has an α^+-activity enabled. Denote the situation where m components $C_1, \ldots, C_m$ synchronise over an α^+-transition by

$$C := C_1 ||_{\{\alpha^+\}} C_2 ||_{\{\alpha^+\}} \cdots ||_{\{\alpha^+\}} C_m.$$

Now, the model of the sensor network can be given as in Fig. V.11, where the cluster i is modelled by component C_i. For subsequent considerations we also need a target quantity in the form of $\mathbb{P}(C \in A)$, where $A = \times_{i=1}^{m} A_i$ is a subset of the state space of C. In the spirit of the 2-LDS for common PEPA models we require this set to contain either only global non-synchronising or only global synchronising states.

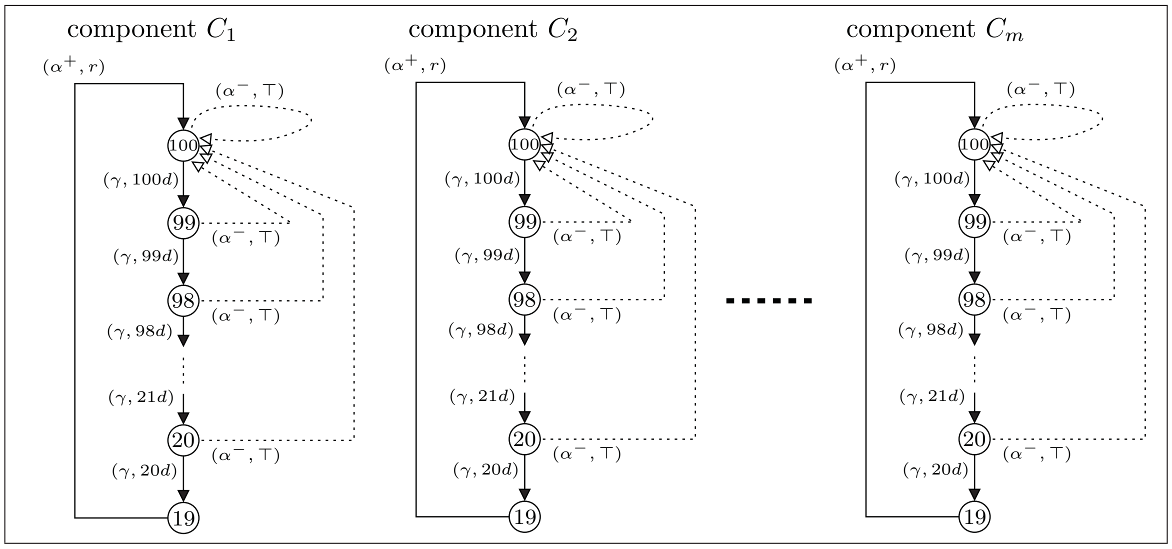


Figure V.11: Sensor network. Synchronisation is initialised as soon as one of the components enters its local state 19. Then, the α^+-activity of that component forces all other components to participate in the synchronisation over an α^--activity.

V.2.3.1 The Embedded DTMC

In order to construct the embedded DTMC the set S of global synchronising states and the set NS of global non-synchronising states which can be reached immediately after a global synchronisation have to be identified. From the target states of the α^+- and α^--activities we see that a global synchronisation always leads to the successor state $(100, 100, \ldots, 100)$. Hence, we have $NS = \{(100, 100, \ldots, 100)\}$. A global synchronising state is reached if one of the $m - 1$ components reaches the state 19. Then, the synchronisation is initiated immediately, no matter which states the other components currently occupy. Note, that this also denies the possibility that several components are in their local state 19 at the same time. As a consequence, every global state, where exactly one marginal entry is 19, is a synchronising state. For example, the state $(19, 99, 99, 99, \ldots, 99)$ is a synchronising state. Of course, depending on the value of m this can lead to a large number of global synchronising states, hence, a large embedded DTMC. But since all global synchronising states lead to the same successor state with the same rate, all global synchronising states can be aggregated into a single state, say $\mathbf{s}$. Finally, we have $S = \{\mathbf{s}\}$ and $NS = \{(100, 100, \ldots, 100)\}$. As is easily verified, the embedded DTMC alternates between the two states $\mathbf{s}$ and $(100, 100, \ldots, 100)$, where the transition probabilities between the two states is 1 each.

A general algorithm to establish the embedded DTMC from an MPA model with preemptive synchronisations would certainly include an exploration procedure which identifies embedded (i.e., global synchronising) states. It could operate in loose analogy to the algorithm to establish the embedded DTMC in the 2-LDS (section III.5). In addition to that the algorithm should include a mechanism to aggregate suitable global synchronising states.

V.2.3.2 The Subsystems

The subsystem $SUB(\mathbf{s})$ which is associated with the global synchronising state $\mathbf{s}$ consists only of the state $\mathbf{s}$, which of course makes it easy to solve. In general, the solution of a subsystem $SUB(\mathbf{x})$, with $\mathbf{x} \in S$, is exactly the same as in the 2-LDS. The mean sojourn time $T(\mathbf{x})$ of $SUB(\mathbf{x})$ and the expected total time $T(\mathbf{x})(A)$ which $SUB(\mathbf{x})$ spends in the set A are calculated via:

2-LDS(P): The Subsystems – Subprocedure 1

Let $\lambda(\mathbf{x})$ be the total rate out of the synchronising state $\mathbf{x} \in S$. Then

$$T(\mathbf{x}) = \frac{1}{\lambda(\mathbf{x})}.$$

Furthermore, it can immediately be stated that

$$T(\mathbf{x})(A) = \begin{cases} 0 & \text{if } \mathbf{x} \notin A \\ \frac{1}{\lambda(\mathbf{x})} & \text{if } \mathbf{x} \in A \end{cases}.$$

In order to derive a solution for a subsystem $SUB(\mathbf{x})$, with $\mathbf{x} \in NS$ consider the concrete subsystem $SUB((100, 100, \ldots, 100))$ of our example. It needs to be solved for the mean sojourn time $T((100, 100, \ldots, 100))$ and the mean time $T((100, 100, \ldots, 100))(A)$ it spends in the set A.

In order to solve this subsystem we apply a similar technique as int the 2-LDS. In all marginal components set the local states which initiate a global synchronisation (i.e., local states with outgoing α^+-activities) as absorbing states. In addition to that ignore (i.e., remove) all α^--activities. The initial state of each component is set to 100 – this corresponds to the situation that the subsystem $SUB((100, 100, \ldots, 100))$ has just been entered. The time instant where in the modified components the first of the components becomes absorbed corresponds to the time instant where the global synchronising state is entered in the original process. In general a subsystem $SUB(\mathbf{x})$, with $\mathbf{x} \in NS$ is solved by:

2-LDS(P): The Subsystems – Subprocedure 2

Let Y_i be the absorbing Markov chain underlying the modified (absorbing) component C_i, and let $Y := (Y_1, \ldots, Y_m)$ be the absorbing joint CTMC, with initial state $\mathbf{x} = (x_1, \ldots, x_m) \in NS$.

- $T(\mathbf{x})$ is the mean time time until **the first** of the marginal CTMCs of $Y := (Y_1, \ldots, Y_m)$ becomes absorbed. That means, it is the expected total time which Y spends in the set $\times_{i=1}^{m} \Theta_i$, where Θ_i contains all transient states of Y_i.
- $T(\mathbf{x})(A)$ is the expected total time Y spends in A before **the first** of the marginal CTMCs becomes absorbed. That means:
  (a) if A contains only non-synchronising states: $T(\mathbf{x})(A)$ is the expected total time Y spends in A.
  (b) if A contains only synchronising states: $T(\mathbf{x})(A) = 0$.

We make the following notes concerning (a) and (b):

(a) In the case of pre-emptive synchronisation, the set $A = \times_{i=1}^{m} A_i$ contains only non-synchronising states iff all of the A_i contain only transient states.

(b) The state $\mathbf{x} = (x_1, \ldots, x_m)$ is synchronising iff at least one of the x_i is a local synchronising state, i.e., a local state with an outgoing α^+-activity.

V.3 Conclusion

In this chapter two variants of the 2-LDS which is applicable to compute steady-state probabilities of cyclic PEPA models were sketched:

2-LDS(T). Instead of cyclic PEPA models, i.e., models which possess an ergodic underlying CTMC, we considered terminating PEPA models. Termination can either occur in the form of a deadlock, or if the system evolves into the predefined component *stop* in which it ceases to execute any activities by definition. The extension of the PEPA language by the component *stop* was proposed in [44]. Since the models under consideration are terminating, the measures of interest are transient measures (instead of steady-state probabilities). The decomposition of the considered PEPA model is almost identical to the case of the 2-LDS. The model is decomposed along the set of embedded states $E^{(X)}$ into subsystems and an embedded DTMC (first level of compositionality), and the solution of the subsystems can be obtained in a compositional way (second level of compositionality). The 2-LDS(T) differs from the 2-LDS in two details which are both due to the fact that for terminating processes only transient measures are of interest. (a) Recall that every embedded state defines a subsystem. In the case of the 2-LDS, where steady-state behaviour is investigated, it can be assumed that the overall system starts in a state contained in S (set of global synchronising states) or NS (set of states to be reached immediately after a synchronisation), hence, in the case of the 2-LDS we have $E^{(X)} = S \cup NS$. In the case of the 2-LDS(T) this assumption is not valid, since transient measures depend on the initial state of the system. Hence, in the 2-LDS(T) the initial state needs an extra treatment. (b) The second difference concerns the embedded DTMC. In the case of 2-LDS the embedded DTMC had to be solved for its steady-state distribution. In the case of the 2-LDS(T), where only transient measures are of interest, the embedded DTMC is solved for visit counts.

2-LDS(P). This variant of the two-level decomposition scheme is applicable to MPA models, where the common barrier synchronisation is replaced by a mechanism which we refer to as pre-emptive synchronisation. In a pre-emptive synchronisation a process which becomes ready to synchronise immediately initiates the synchronisation, i.e., all other processes which are to take part in the synchronisation, are forced to do so immediately. The measures of interest are steady-state probabilities of the overall system. Again, this method consists of two levels of compositionality. In complete analogy to the 2-LDS the system is decomposed into subsystems and an embedded DTMC on the first level of compositionality. The second level of compositionality is a bit different. In the 2-LDS subsystems were represented as the parallel evolution of several independent absorbing CTMCs, where time instants of joint absorption corresponded to time instants, at which the subsystems were left. In the case of the 2-LDS(T) subsystems can still be represented as the parallel evolution of several independent absorbing CTMCs. But now the time instant, at which a subsystem is left, corresponds to the time instant, where the first of the marginal CTMCs become absorbed. Despite their slightly different structure, the subsystems generated by the 2-LDS(T) can be solved with the same methods as the subsystems generated by the 2-LDS.

Further possible extensions. All three variants (2-LDS, 2-LDS(T), 2-LDS(P)) of the two-level decomposition scheme share the characteristics that (a) they decompose the model under investigation into subsystems and an embedded DTMC which describes the dynamics between the subsystems, and (b) they yield subsystems which are either trivial, i.e., they possess only a single state, or, if they are non-trivial, can be represented as the parallel composition of several independent absorbing CTMCs. The differences in the three variants stem from different properties of the considered models. If the model is terminating, as opposed to be in steady-state, then only transient measures can be derived and the embedded DTMC must be solved for visit counts instead of the steady-state distribution. A model which possesses pre-emptive synchronisations generates subsystems which differ in structure from subsystems generated by models with barrier synchronisations. Classify subsystems of the former kind as TYPE(P) and subsystems of the latter kind as TYPE(0). Both types of subsystems (if they are non-trivial) can be represented as the parallel composition of several absorbing CTMCs, but different measures of the joint CTMC must be computed, in order to solve the subsystems.

Since subsystems of either kind are solved in isolation, it seems possible that the two-level decomposition scheme can be generalised to be applicable to models which possess both barrier and pre-emptive synchronisations, if the resulting subsystems can unambiguously be classified as either TYPE(0) or TYPE(P). Below, we give an outline of such a decomposition procedure for the case of a model C in steady-state. The target quantity is the steady-state probability $\mathbb{P}(C \in A)$ that the model is in the set of states A.

A Generalised Two-Level Decomposition Scheme: Outline

(Part 1) Make sure that the composite PEPA component C meets certain general requirements which ensure the existence of a steady-state solution and the exploitability of the two levels of compositionality. Furthermore, assure that subsystems generated by this model are either of TYPE(0) or TYPE(P).

(Part 2) Calculate the steady-state distribution π of the embedded DTMC X which is defined by $E^{(X)} = S \cup NS$.

(Part 3) For every subsystem $SUB(\mathbf{x})$, $\mathbf{x} \in E^{(X)}$, calculate the mean sojourn time $T(\mathbf{x})$. In addition, calculate the expected total time spent in the set A, i.e., $T(\mathbf{x})(A)$.

if $SUB(\mathbf{x})$ **is of TYPE(0):** Employ solution method for subsystems of the 2-LDS.

if $SUB(\mathbf{x})$ **is of TYPE(P):** Employ solution method for subsystems of the 2-LDS(P).

Finally evaluate:

$$\mathbb{P}(C \in A) = \frac{\sum_{\mathbf{x} \in E^{(X)}} \pi(\mathbf{x}) T(\mathbf{x})(A)}{\sum_{\mathbf{x} \in E^{(X)}} \pi(\mathbf{x}) T(\mathbf{x})}.$$

Chapter VI

Conclusion

The initial motivation of the current work was the need to antagonise the problem of state space explosion which is inherent to Markovian Process Algebra models. We focused on the computation of single steady-state probabilities of MPA models, where the MPA language PEPA was chosen to illustrate our ideas. A translation of the developed solution technique to other MPAs, however, should not present a severe problem. This work delivers the following three main results which are briefly sketched below:

- The two-level decomposition scheme for MPA models (2-LDS),
- The method of Compositional Uniformisation,
- First steps towards the generalisation of the 2-LDS.

Development of the two-level decomposition scheme (2-LDS) for MPA models. The two-level decomposition scheme is a direct extension of the work in [5] and [6], where the authors develop an algorithm to compute the distribution of a semi-Markov chain which is embedded in the MPA model under investigation. A requirement for this method is that all concurrent components within the MPA model are only allowed to participate in barrier synchronisations. That means, whenever a global synchronisation takes place, all concurrent components must participate in the synchronisation. We must adopt this requirement, but extend the method by the ability to compute single steady-state probabilities of the MPA model. Our approach introduces two levels of compositionality. On the first level the model is decomposed into several subsystems and an embedded DTMC, where a subsystem is some kind of stochastic process subordinated to the original system. Typically the embedded DTMC possesses a relatively small state space, hence, its steady-state distribution can easily be obtained. It is shown that under certain circumstances the subsystems can be represented as the parallel composition of several absorbing

CTMCs; this is referred to as the second level of compositionality. Due to the compositional character of the subsystems, they can efficiently be *solved* for certain cumulative measures. Combined with the solution of the embedded DTMC the individual solutions of the subsystems yield the desired steady-state probability of the overall system.

A compositional solution technique for cumulative measures of absorbing joint Markov chains. The 2-LDS generates subsystems of an MPA model which can be represented as the parallel composition of several absorbing CTMCs, i.e., the joint CTMC of several independent marginal absorbing CTMCs. In this work we presented some well-known methods which can be applied to solve the joint absorbing CTMC for the mean time to absorption, as well as expected total times spent in a subset of the state space. Furthermore, we succeeded in finding a novel approach called *Compositional Uniformisation* to solve the joint CTMC which is able to compete with the known methods with respect to computation time and space requirements.

Towards a Generalisation of the 2-LDS. We widened the class of models which are accessible by modified versions of the 2-LDS. We showed how transient measures of terminating PEPA processes can be obtained with the so called variant 2-LDS(T). The second variant 2-LDS(P) is applicable to MPA models in steady-state, where the barrier-type synchronisation is replaced by a mechanism we refer to as pre-emptive synchronisation. In a pre-emptive synchronisation a component which becomes ready to synchronise immediately initiates the synchronisation, i.e., a component is never waiting for other components to become ready to synchronise. Both variants of the 2-LDS, however, still require that all sequential components are involved in every synchronisation.

Future Prospects. When thinking about possible improvements of our approach, the first thing that comes to mind will certainly be the elimination of the severe requirements which we impose on the model under investigation. The first requirement is the fact that we can only treat models in which all sequential components are involved in every synchronisation. As a second constraint, we can only compute single state probabilities of the model, but not its entire distribution at once. Future research will probably center around the first of these two aspects, although at the moment we can only guess if such an attempt would be fruitful.

Bibliography

[1] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.

[2] Falko Bause and Peter Buchholz, editors. *Proceedings 14th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB 2008), March 31 - April 2, 2008, Dortmund, Germany*. VDE Verlag, 2008.

[3] M. Bernardo, L. Donatiello, and R. Gorrieri. Modeling and analyzing concurrent systems with mpa. In *Proc. of the 2nd Process Algebra and Performance Modeling Workshop,(Erlangen, July 1994)*, volume 27, FAU Erlangen-Nürnberg, Germany, 1994.

[4] Marco Bernardo and Roberto Gorrieri. Extended Markovian process algebra. In *CONCUR '96: Proceedings of the 7th International Conference on Concurrency Theory*, pages 315–330, London, UK, 1996. Springer-Verlag.

[5] Henrik Bohnenkamp. *Compositional Solution of Stochastic Process Algebra Models*. PhD thesis, Department of Computer Science, Rheinisch-Westfälische Technische Hochschule Aachen, Germany, 2002.

[6] Henrik C. Bohnenkamp and Boudewijn R. Haverkort. The mean value of the maximum. In *PAPM-PROBMIV '02: Proceedings of the Second Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, pages 37–56, London, UK, 2002. Springer-Verlag.

[7] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. Wiley-Interscience, New York, NY, USA, 1998.

[8] R. J. Boucherie. A characterization of independence for competing Markov chains with applications to stochastic petri nets. *IEEE Trans. Softw. Eng.*, 20(7):536–544, 1994.

[9] Freimut Brenner. Absorbing joint markov chains: Exploiting compositionality for cumulative measures. In Khalid Al-Begain, Armin Heindl, and Miklós Telek, editors, *Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*, pages 137–145, 2007.

[10] Freimut Brenner. Cumulative measures of absorbing joint Markov chains and an application to Markovian process algebras. Technical Report 12, Institute for Computer Science and Business Information Systems, University of Duisburg-Essen, NOV 2007.

[11] Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors. *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[12] Peter Buchholz. Compositional analysis of a Markovian process algebra. In U. Herzog and M. Rettelbach, editors, *Proceedings of the 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[13] Mariacarla Calzarossa and Salvatore Tucci, editors. *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, London, UK, 2002. Springer-Verlag.

[14] E. Çinlar. *Introduction to Stochastic Processes*. Prentice-Hall, New Jersey, USA, 1975.

[15] Gianfranco Ciardo, Gerald Lüttgen, and Andrew S. Miner. Exploiting interleaving semantics in symbolic state-space generation. *Form. Methods Syst. Des.*, 31(1):63–100, 2007.

[16] P. J. Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, New York, NY, USA, 1977.

[17] Mor Harchol-Balter, Marta Kwiatkowska, and Miklos Telek, editors. *Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST)*. IEEE, September 2007.

[18] Günter Haring, Christoph Lindemann, and Martin Reiser, editors. *Performance Evaluation: Origins and Directions*, London, UK, 2000. Springer-Verlag.

[19] Peter Harrison and Jane Hillston. Exploiting quasi-reversible structures in Markovian process algebra models. *The Computer Journal*, 38(7):510–520, 1995.

[20] H. Hermanns. *Interactive Markov Chains and the Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[21] H. Hermanns and M. Rettelbach. Syntax, semantics, equivalences, and axioms for mtipp. In *Proc. of PAPM '94, Erlangen, Germany*, pages 71–87, 1994.

[22] Holger Hermanns, Joachim Meyer-Kayser, and Markus Siegle. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *3rd International Workshop on the Numerical Solutions of Markov Chains, NSMC 99, Zaragoza (Spain)*, pages 188–207, 1999.

[23] Ulrich Herzog. Formal description, time and performance analysis. a framework. In *Entwurf und Betrieb verteilter Systeme, Fachtagung der Sonderforschungsbereiche 124 und 182*, pages 172–190, London, UK, 1990. Springer-Verlag.

[24] J. Hillston. A class of PEPA models exhibiting product form solution over submodels. Technical Report LFCS-98-382, LFCS, Department of Computer Science, University of Edinburgh, 1998.

[25] J. Hillston and N. Thomas. A syntactical analysis of reversible PEPA models. In C. Priami, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998.

[26] Jane Hillston. The nature of synchronisation. In *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, pages 51–70, 1994.

[27] Jane Hillston. *A compositional approach to performance modelling.* Cambridge University Press, New York, NY, USA, 1996.

[28] Jane Hillston. Exploiting structure in solution: decomposing compositional models. pages 278–314, 2002.

[29] Jane Hillston. Tuning systems: From composition to performance (the needham lecture). *The Computer Journal*, 48(4):385–400, 2005.

[30] Jane Hillston and Vassilis Mertsiotakis. A simple time scale decomposition technique for stochastic process algebras. *The Computer Journal*, 38(7):566–577, 1995.

[31] Jane Hillston and Nigel Thomas. Product form solution for a class of PEPA models. *Perform. Eval.*, 35(3-4):171–192, 1999.

[32] W. Hoeffding. Probability inequalities for sums of bounded random variables. *j-J-AM-STAT-ASSOC*, 58(301):13–30, March 1963.

[33] Matthias Kuntz. *Symbolic Semantics and Verification of Stochastic Process Algebras.* PhD thesis, Institut für Informatik 7, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2006.

[34] Kai Lampka. *A symbolic approach to the state graph based analysis of high-level Markov Reward Models.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2007.

[35] Zhen Liu, Vishal Misra, and Prashant J. Shenoy, editors. *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2008, Annapolis, MD, USA, June 2-6, 2008.* ACM, 2008.

[36] Vassilis Mertsiotakis. Time scale decomposition of stochastic process algebra models. In E.Brinksma and A. Nymeyer, editors, *Proc. of 5th Process Algebra and Performance Modelling Workshop*, 1997.

[37] Vassilis Mertsiotakis. Approximate analysis methods for stochastic process algebras. (phd-thesis). Technical Report 31-10, University of Erlangen, 1998.

[38] Andrew S. Miner and David Parker. Symbolic representations and analysis of large probabilistic systems. In *Validation of Stochastic Systems*, pages 296–338, 2004.

[39] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[40] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. The Johns Hopkins University Press, Baltimore/London, 1981.

[41] Brigitte Plateau and Karim Atif. Stochastic automata network for modeling parallel systems. *IEEE Trans. Software Eng.*, 17(10):1093–1108, 1991.

[42] M. L. Rettelbach and Markus Siegle. Compositional minimal semantics for the stochastic process algebra tipp. In U. Herzog and M. Rettelbach, editors, *Proceedings of the 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[43] N. Thomas and S. Gilmore. Applying quasi-separability to Markovian process algebra. In C. Priami, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998.

[44] Nigel Thomas and Jeremy Bradley. Terminating processes in PEPA. In *UKPEW'01: Proceedings of 17th Annual UK Performance Engineering Workshop*, pages 143–154, 2001.

Appendices

Appendix A: The Relation $\mathbb{E}\left[H^{Y_2}_{\beta(H^{Y_1}_\alpha)}\right] = \mathbb{E}\left[H^{Y_2}_{\beta'}\right]$

Let T be matrix which contains the transition rates between transient states of Y_2. Then, according to (IV.2) the mean time to absorption of Y_2, with initial distribution β, is given by

$$\mathbb{E}\left[H^{Y_2}_{\beta}\right] = (-1)\beta T^{-1}1. \tag{A.1}$$

In order to determine the expectation $\mathbb{E}\left[H^{Y_2}_{\beta(H^{Y_1}_\alpha)}\right]$, one has to be aware that $\beta(H^{Y_1}_\alpha)$ is a random variable itself. That means, equation (A.1) must be extended such that $\beta(H^{Y_1}_\alpha)$ is deconditioned. Let $F^{Y_1}(t)$ be the distribution function of $H^{Y_1}_\alpha$. Then

$$\mathbb{E}\left[H^{Y_2}_{\beta(H^{Y_1}_\alpha)}\right] = \int_0^\infty -1\beta(t)T^{-1}1dF^{Y_1}(t) = \left[\underbrace{\int_0^\infty \beta(t)dF^{Y_1}(t)}_{=\mathbb{E}\left[\beta(H^{Y_1}_\alpha)\right]}\right](-1)T^{-1}1.$$

That means $\mathbb{E}\left[H^{Y_2}_{\beta(H^{Y_1}_\alpha)}\right]$ is the mean time to absorption of Y_2 if the initial distribution $\mathbb{E}\left[\beta(H^{Y_1}_\alpha)\right]$ is used. With $\beta' = \mathbb{E}\left[\beta(H^{Y_1}_\alpha)\right]$, we obtain

$$\mathbb{E}\left[H^{Y_2}_{\beta(H^{Y_1}_\alpha)}\right] = \mathbb{E}\left[H^{Y_2}_{\beta'}\right].$$

Appendix B: A Word on Binomial Probabilities

Let $b(\cdot; n, p)$ be a binomial distribution with

$$b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k} \quad \text{and} \quad 0 < p < 1.$$

Assume $0 \leq k \leq n$. Then the following is trivial:

$$\begin{aligned} b(k; n, p) &= b(k-1; n, p)\frac{(n-k+1)p}{k(1-p)} = b(k-1; n-1, p)\frac{np}{k}, \quad \text{for } k \geq 1, \\ b(k; n, p) &= b(k+1; n, p)\frac{(k+1)(1-p)}{(n-k)p} = b(k; n-1, p)\frac{n(1-p)}{n-k}, \quad \text{for } k \leq n-1. \end{aligned} \tag{B.1}$$

The mode of $b(\cdot; n, p)$ is given by $M := \lfloor (n+1)p \rfloor$. For the modes M and M' of $b(\cdot; n, p)$ and $b(\cdot; n+1, p)$, we have

$$M' - M \in \{0, 1\}.$$

When computing binomial probabilities, one might want to avoid to compute extremely low probabilities. Fortunately, for a given $\gamma > 0$, it is possible to determine values ℓ_n and r_n in advance, such that $\sum_{k=\ell_n}^{r_n} b(k; n, p) \geq 1 - \gamma$. For example, to find the truncation index ℓ_n for the lower tail, consider a random variable $X \sim b(\cdot; n, p)$ and the Chernoff bound (see [39])

$$P(X < (1-\delta)np) \leq e^{-\delta^2 np/2} \qquad \text{or equivalently} \qquad P(X < \ell_n) \leq e^{-\frac{(np-\ell_n)^2}{2np}}, \tag{B.2}$$

for $\delta > 0$ and $\ell_n < np$.

In order to compute r_n, note that $b(k; n, p) = b(n-k; n, 1-p)$. With a random variable $X' \sim b(\cdot; n, 1-p)$ we obtain, for $r_n + 1 > np$,

$$P(X > r_n) = P(X' < n - (r_n + 1)).$$

Lets say $\gamma = \alpha + \beta$. Then

$$\ell_n = \max\left\{0, \left\lfloor np - \sqrt{-2np \ln \alpha} \right\rfloor\right\} \qquad \text{and} \qquad r_n = \min\left\{n, \left\lceil np + \sqrt{-2n(1-p)\ln\beta} - 1 \right\rceil\right\} \tag{B.3}$$

satisfy $P(\ell_n \leq X \leq r_n) > 1 - \gamma$, or equivalently

$$1 - \sum_{k=\ell_n}^{r_n} b(k; n, p) \leq \gamma.$$

Other bounds in the style of (B.2) exist and can be used as well to determine ℓ_n and r_n.

E.g., Hoeffding's inequality ([32]) yields the bound $P(X \leq \ell_n) \leq e^{-2\frac{(np-\ell_n)^2}{n}}$, for $\ell_n < np$, which apparently is a tighter bound than (B.2) if $p > 1/4$.

With these introductory explanations the following algorithm is almost self-explaining. For given numbers ℓ_n and r_n, the n-th pass of the outer loop computes the binomial probabilities $b(k; n, p)$, $k = \ell_n \dots r_n$, according to (B.1). The number n is bounded by $0 \leq n \leq N$, where N is given by the application which uses the binomial probabilities. In order to reduce the effect of possible underflow, the algorithm starts with the computation of the mode M' and its corresponding binomial probability $b(M'; n, p)$, for each n.

```
**Binomial Probabilities:**
$b(0; 0, p) = 1$;
For $n = 1 \dots N$ do:

    $M = \lfloor np \rfloor$;
    $M' = \lfloor (n+1)p \rfloor$;
    if($M' > M$)
        $b(M', n, p) = b(M, n-1, p)\frac{np}{M'}$;
    else
        $b(M', n, p) = b(M, n-1, p)\frac{n(1-p)}{n-M'}$;
    For $k = M+1 \dots r$ do:    $b(k; n, p) = b(k-1; n, p)\frac{(n-(k-1))p}{k(1-p)}$;
    For $k = M-1 \dots \ell$ do:    $b(k; n, p) = b(k+1; n, p)\frac{(k+1)(1-p)}{(n-k)p}$;
    – use $b(\ell; n, p), \cdots, b(r; n, p)$ in application –
    store $b(M'; n, p)$; delete all other $b(\cdot; n, p)$;
```

From (B.3) it is seen that, for every fixed positive γ, $r_n - \ell_n \in O(\sqrt{n})$. We assume γ to be fixed (or at least bounded), since in practise it would be chosen within machine precision. Hence, the $n-$th pass of the outer loop requires time $O(\sqrt{n})$. The time complexity of the overall algorithm is in

$$O(N^{3/2}).$$

Appendix C: Notation for Chapter IV

Marginal Absorbing Markov Chains $i \in \{1, \dots, m\}$:

Y_i	absorbing Markov chain
E_i	state space of Y_i
$S_i \subset E_i$	set of absorbing states of Y_i
Q_i	generator matrix of Y_i
$q_i \geq \max_{j \in E_i}\{\lvert Q_i(j,j)\rvert\}$	uniformisation rate of Y_i
$P_i := I + 1/q_i Q_i$	(uniformised) one-step transition matrix
$\nu_i(n) = \nu_i(0)P_i^n = \nu_i(n-1)P_i$, for $n \geq 1$	distribution of the uniformised CTMC Y_i at the $n-$th step, with the initial distribution $\nu_i(0)$
$\mathcal{A}_i \subseteq E_i$	some subset of E_i
$A_i \subseteq E_i$	some subset of E_i
$\nu_i(n)(\mathcal{A}_i) := \sum_{x \in \mathcal{A}_i} \nu_i(n)(x)$	aggregated state probability at the $n-$th step
$\nu_i[\mathcal{A}_i](n) := \nu_i(n)(\mathcal{A}_i)$	$\nu_i[\mathcal{A}_i](n)$ is interpreted as a function of n

Absorbing Joint Markov Chain

$Y = (Y_1, \dots, Y_m)$	joint absorbing Markov chain
$E := \times_{i=1}^m E_i$	state space of Y
$S := \times_{i=1}^m S_i$	set of absorbing states of Y
$Q = \oplus_{i=1}^m Q_i$	generator matrix of Y
$q := q_1 + \cdots + q_m$	uniformisation rate of Y
$P = I + \frac{1}{q}Q$	(uniformised) one-step transition matrix of Y
$\nu(n) = \nu(0)P^n = \nu(n-1)P$, for $n \geq 1$	distribution of the uniformised CTMC Y at the $n-$th step, with the initial distribution $\nu(0)$
$\mathcal{A} := \times_{i=1}^m \mathcal{A}_i \subseteq E$	a certain subset of E
$A := \times_{i=1}^m A_i \subseteq E$	a certain subset of E
$\nu(n)(\mathcal{A}) = \sum_{x \in \mathcal{A}} \nu(n)(x)$	aggregated state probability of the (uniformised) CTMC Y at the $n-$th step
$\nu[\mathcal{A}](n) := \nu(n)(\mathcal{A})$	$\nu[\mathcal{A}](n)$ is interpreted as a function of n