Approaches to Analysis and Simplification of non-Markovian System Models

Fida Kamal Dankar

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in Computer Science

School of Information Technology and Engineering (SITE)
Faculty of Engineering
University of Ottawa

Abstract

In this thesis, we present an algorithm to transform a subset of generalized semi-Markov processes into semi-Markov processes. The transformation preserves steady-state simulation, a simulation that allows us to retrieve the steady state probability of the generalized semi-Markov process from that of the transformed process. The method presented could generate semi-Markov processes with big state spaces, for that reason we introduce a two state simplification techniques. The first one deals with the state space explosion problem by deleting states from the original generalized semi-Markov process. The aim of this technique is to generate semi-Markov processes with smaller state space. The technique deletes states from the generalized semi-Markov process while preserving the distribution of time needed to travel between non-deleted states; the technique also preserves the transient state probabilities of a subset of the states in the process. The other technique deals with the state space explosion problem at the level of semi-Markov processes. It works by deleting states from the semi-Markov processes while preserving the average time to travel between non-deleted states, or what we call mean passage-time equivalence, the technique also preserves the steady state probabilities of a subset of the states in the process.

Table of Contents

Table of Contents

ABSTRACT .... **II**

TABLE OF CONTENTS .... **III**

LIST OF FIGURES .... **VII**

LIST OF ABBREVIATIONS .... **IX**

ACKNOWLEDGMENT .... **XI**

CHAPTER 1. INTRODUCTION **1**

1.1. MOTIVATION 1

1.2. THESIS CONTRIBUTIONS 5

1.3. OUTLINE OF THE THESIS 6

CHAPTER 2: PERFORMANCE MODELING **8**

2.1. INTRODUCTION 8

2.2. MODELS 9

2.3. STOCHASTIC MODELS 11

2.3.1. Exponential Models *14*

2.3.2. Non Exponential Models *17*

2.3.2.1. Semi-Markov processes and generalized semi-Markov processes 18

A. SMP definition. 19

B. Embedded Markov chain 20

C. Example of an SMP. 21

D. Transient and steady probabilities for semi-Markov models 22

2.3.2.2. Generalized semi-Markov process 24

A. Formal definition and examples 24

B. Transient and steady state probabilities for GSMPs 27

2.4. QUEUING MODELS 29

2.4.1. Introduction *29*

2.4.2. Single Queues *29*

2.4.3. Networks of Queues *31*

2.4.5. Product Form Queuing Networks *32*

2.4.5. Expressiveness *33*

2.4.6. Queuing Networks: Pros and Cons *33*

2.5. PETRI NETS 33

2.5.1. General Notions *34*

2.5.2. Petri Nets: Pros and Cons *36*

2.6. PROCESS ALGEBRAS 37

2.6.1. PEPA *38*

2.6.2. Non-Markovian Process Algebras *40*

2.6.3. Process Algebras: Pros and Cons *41*

CHAPTER 3: NON-MARKOVIAN ANALYSIS 43

3.1. ANALYSIS APPROACHES FOR NON-MARKOVIAN MODELS 43

3.2. METHOD OF SUPPLEMENTARY VARIABLES 44

3.3. METHOD OF EMBEDDED REGENERATIVE PROCESSES 46

3.4. CONTINUOUS PHASE TYPE DISTRIBUTION 49

3.5. CONTRIBUTIONS OUTLINE 49

CHAPTER 4: A DEEPER LOOK INTO NON-MARKOVIAN ANALYSIS 52

4.1. INTRODUCTION 52

4.2. PRELIMINARY DEFINITIONS AND ALGORITHM OVERVIEW 55

4.2.1. Preliminary definitions..*56*

4.2.2. Algorithm overview *58*

4.3. DEFINITIONS, ILLUSTRATIONS AND PRELIMINARY RESULTS 64

4.3.1. General Definitions and Results *64*

4.3.2. Properties of NRGSMP. *74*

4.3.3. HMRP Definition and Properties..*74*

4.3.4 Properties of Average Residual Times...*80*

4.3.5. Transient and Steady-State Simulations: Definitions and Properties...............*87*

4.4. FROM NRGSMP TO SMP..94

4.4.1. Algorithm 1: NRGSMP to HMRP *94*

4.4.1.1. Overview 94

4.4.1.2. Algorithm 1...97

4.4.1.3. Definition of the major functions 100

4.4.2. HMRP to SMP..*104*

4.4.2.1. Overview 104

4.4.2.2. Algorithm 2...105

4.4.3. Application..*106*

4.5. PRACTICAL LIMITATIONS 107

4.5.1. Space Complexity..*107*

4.5.2. Time Complexity..*111*

4.6. CONCLUSION 114

CHAPTER 5: TIME PRESERVING SIMPLIFICATION FOR GSMPS....................116

5.1. INTRODUCTION 116

5.2. DEFINITION OF EQUIVALENCES 117

5.3. SIMPLIFICATION TECHNIQUE 118

5.3.1. Illustration..*121*

5.3.2. Overall Algorithm *123*

5.3.3. Complexity *126*

CHAPTER 6: MEAN PASSAGE-TIME EQUIVALENCE FOR SMPS 127

6.1. INTRODUCTION 127

6.2. EQUIVALENCE DEFINITION 128

6.3. SIMPLIFICATION TECHNIQUE 129

6.3.1. Basic Reduction Steps *129*

6.3.2. Algorithm *132*

6.4. COMPLEXITY 134

6.5. EFFECTS ON THE ORIGINAL NRGSMP 135

CHAPTER 7: ILLUSTRATIONS AND APPLICATIONS 137

7.1. A SIMPLE EXAMPLE 137

7.1.1. NRGSMP Simplification *139*

7.1.2. NRGSMP to HMRP and then to SMP *140*

7.1.3. SMP Simplification *140*

7.2. PROPERTIES OF NRGSMPS 143

7.3. CASE STUDY 144

7.3.1. Software Rejuvenation *144*

7.3.2. Extension of the Periodic Rejuvenation Model *147*

7.3.3. Analysis of the Periodic Rejuvenation Model *150*

7.3.4. Analysis of the Periodic Rejuvenation with Back-up Model *155*

CHAPTER 8: CONCLUSION AND FUTURE DIRECTIONS 157

REFERENCES 161

List of Figures

Figure 1. Concurrent events 4

Figure 2. An example of a semi-Markov process 22

Figure 3. Example of a GSMP 26

Figure 4. G/G/1/ 27

Figure 5. Two GSMPs....55

Figure 6. Part of a GSMP....57

Figure 7. Nest....66

Figure 8. An example of a single-AvRes-set 72

Figure 9. An HMRP 76

Figure 10. Example of an HMRP....76

Figure 11. Two consecutive states 81

Figure 12. SMPs $N(0-0)$ and $\vec{N}(0-0)$ 107

Figure 13. Sequential reduction 120

Figure 14. Average sequential reduction 130

Figure 15. Alternate reduction 131

Figure 16. Cycle removal....132

Figure 17. Example of an NRGSMP....................138
Figure 18. ESMP....................139
Figure 19. The sub-SMP....................141
Figure 20. Deletion of State 1-1....................142
Figure 21. Periodic rejuvenation....................147
Figure 22. Periodic rejuvenation with backup unit....................148
Figure 23. HMRP for the periodic rejuvenation with back-up. 156

List of Abbreviations

AvRes	Average Residual distribution
CTMC	Continuous Time Markov Chain
DES	Discrete Event Simulation
EGSMP	Enabling restriction Generalized Semi-Markov Process
ESMP	Embedded Semi-Markov Process
GSMP	Generalized Semi-Markov Process
HMRP	Hidden Markov Regenerative Process
MRGP	Markov ReGenerative Process
MTTA	Mean Time To Absorption
MTTF	Mean Time To Failure

MTTR	Mean Time To Repair
NM	Near-Markovian
NRGSMP	Near Regenerative Generalized Semi-Markov Process
NSM	Near Semi-Markovian
PE	Performance Engineering
REG	Regenerative
SMP	Semi Markov Process
SPA	Stochastic Process Algebra
SPN	Stochastic Petri-Nets
SSP	Steady State Probability
s-simulation	steady state simulation
t-simulation	transient state simulation
TSP	Transient State Probability

Acknowledgment

I would like to express my deep gratitude to my supervisor Gregor v. Bochmann, who spent many hours reading my writings and initiating great discussions. I learned a lot from his analytical skills, and I greatly benefited from his unbounded imagination and his capacity to make every idea more comprehensible through an example, no matter how abstract the idea is. I would like to thank Professor Stefan Haar for his time in discussing and providing feedback on my work. And I would like to also thank my thesis committee for their valuable feedback, especially Professor Joost-Pieter Katoen and Professor Dorina Petriu.

Great thanks to my family. Pour la personne la plus chère à mon cœur, ma mère Najah Ghomrawi pour tous ses sacrifices aux cours des années, pour son amour et pour m'avoir appris d'aimer à apprendre. To my dearest husband Wissam Elcheikh Ali for his love, his patience, his unstoppable encouragement, and for assuming more family responsibility so I would have more time to work. To my two dearest kids Ibrahim and Ahmed who taught me how to balance work and family life, and who showed me the importance of play. To my dear sisters Sana, Mayssoun, Gazoie, Samar and Hala, to my nieces and nephews with whom I share a deep affection. I love you all. Also big thanks to my sister in law Hiba and my mother in law Oum Kassem for their help and for never saying no when I needed them.

And finally to my dear dad Kamal Dankar I would like to dedicate these two sentences in the language he likes and masters:

إلى الذي شجعني على ألمضّي و حفزّني على ألإنجاز... إلى فخري وإعتزازي... الى والدي الغالي كمال دنكر

Translation: To the person who encouraged me to continue and to achieve... To my pride... To my dear dad Kamal Dankar.

إلى حبيبي وسام

To My dear Wissam

Chapter 1. Introduction

1.1. Motivation

Performance engineering (PE) covers modeling, analysis and synthesis of systems. Temporal behavior of real systems is measured and modeled, characteristic performance measures are then defined and measured [16],[52],[71],[76],[82]. The general scenario is as follows:

- The environment generates requests; these are known as the workload to the system: The workload represents the sum of all needed activities and services such as type of activities and frequency of requests.
- The system consists of one or more components trying to satisfy these requests.
- An optimal system structure is reached if the system fulfills all requirements concerning the quality of service such as liveness, throughput and response time.

The steps to achieve PE are:

- Workload characterization and system parameter specification,
- Modeling,
- Analysis, which involves extracting performance measures from the model using methods of statistics, stochastic processes, or simulation.

For more details about the above steps refer to [76].

In this thesis, we focus mainly on modeling. In general, we have two main modeling categories: hard real-time and soft real-time systems:

- Hard real-time systems need deterministic timing models because actions take place at distinct time instants or within fixed time intervals. Examples of hard real time systems are avionic systems and robots. Properties of interest in such systems include safety and liveness. Typical modeling techniques are: timed automata [4], timed Petri nets, timed process algebra.
- Soft real-time systems need stochastic timing models due to contention, faults, and random service strategies. Examples include time sharing computers and telephone systems. Properties of interest in such systems include: throughput, utilization, and delays. Randomly varying time delays are captured by stochastic processes or by high-level models such as stochastic Petri-Nets (SPN), queuing networks or stochastic process algebras (SPA) whose underlying process is a stochastic process. To extract performance parameters from SPN and SPA, we need to generate the underlying stochastic process and analyze it.

The stochastic processes that are mostly studied in the literature for performance and dependability purposes are in increasing order of expressivity: Continuous-Time Markov Chains (CTMC) [25], Semi-Markov Processes (SMP) [25],[56],[59], and Generalized Semi-Markov Processes (GSMP) [67]. The difference between them lies in the set of instants in the process life that satisfy the *Markov property*, A stochastic process satisfies the Markov property at time instant t, if the conditional probability distribution of the states of the process after time instant t is conditionally independent of the states of the process before time t (path of the process), given its state at time t. For CTMCs this property holds at every instant of the process life, for SMPs the property holds at the instants of state change only, and for GSMPs, the property may never hold.

Because of the holding of the Markovian property at all instants of the process life, CTMCs can only represent activities with an exponentially distributed duration [19]. The only candidates for representing systems with generally distributed, not necessarily exponential, event durations are SMPs and GSMPs. SMPs and GSMPs are state automata whose transitions are triggered by the occurrence of stochastically timed events. A set of active events $A(s)$ is associated with each state s of the automaton. Each event has an associated generally distributed lifetime. If in a state s the life of an active event e terminates, one says that event e occurs, and the automata moves to another state. Active events of a state compete to trigger the next transition.

SMPs are not suitable for representing event concurrency, as explained in the following example:

Example 1: Consider two concurrent events *e* and *e'* running in parallel; event *e* has a deterministic duration of 1 time unit, and event *e'* has a geometrically distributed duration with parameter 0.1. The situation is represented in Figure 1. Each state is annotated with a set of events; these are the events that become active once we enter the state (i.e. they are assigned a lifetime according to their distribution once we enter the state, we also say that the events are initialized in that state). In state 0 of the system shown in Figure 1, both events *e* and *e'* are initialized; the system stays in state 0 until the lifetime of one of the events terminates. At that point, the event occurs, and a transition labeled with the event takes the process to a new state. If event *e* occurs first, we move to state 2, the sojourn time in state 2 (which is determined by the residual lifetime of event *e'*) is not given, it rather depends on the time spent in state 0. Hence the above process is not Markovian at the point we enter state 2. So even the simplest case of two concurrent activities can not be represented by an SMP. The example above is in fact a GSMP.

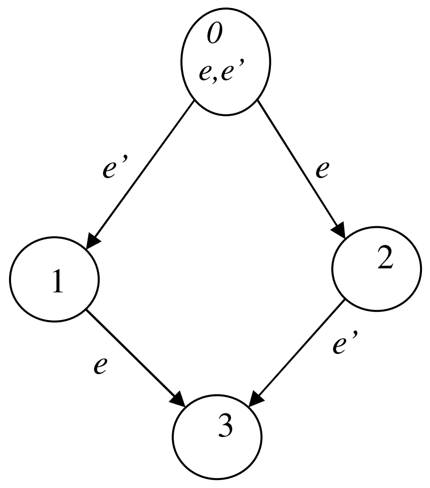


Figure 1. Concurrent events

The Transient State Probability of a stochastic process (TSP) is the probability of being in a given state of the process at a given time, And the steady state probability (SSP) is the probability of being in a given state of the process at equilibrium, i.e. after the system hs been in operation for a pretty long time. The TSPs and SSPs are everything we need to know to be able to extract key performance and dependability measures from a process; for that reason, finding the TSPs and/or SSPs is referred to as "solving the process".

In Markov processes (CTMC, SMP), the calculation of the transient and steady state probabilities is possible through a straightforward application of linear algebra [15],[27],[73],[78].

The absence of the Markov property in GSMPs renders the solution of these systems a tedious task. Since events associated with computer and communication systems may be concurrent and have a distribution of general nature [61], the modeler needs to be provided with tools for quantitative analysis of performance and dependability in GSMPs. The existing approaches that dealt with this problem are limited in applicability to processes whose general events (i.e. non-exponentially distributed events) are mutually exclusive, referred to as EGSMPs.

1.2. Thesis Contributions

As mentioned above, existing approaches that analyze GSMPs are limited in applicability to EGSMPs. In fact, imposing this restriction (the mutual exclusivity of the non-exponential distributions) leads to algorithms with reasonable costs while going beyong the restriction is one of the most challenging open issues in the field.

In this thesis, we will define a technique that relaxes the above restriction by allowing several non-exponentially distributed events to be enabled at any time; however, we impose a restriction on the type of cycles allowed; the GSMPs with the cycle restrictions will be referred to as near-regenerative semi-Markov Processes, NRGSMPs. The NRGSMP's will be presented in Chapter 3, and formally defined in Chapter 4.

In Chapter 4, we will show that the set of EGSMPs is a subset of the set of NRGSMPs; moreover, we it will be proved that the models shown in Figure 3, Figure 10, as well as the software rejuvenation models presented in Section 7.3 are all examples of NRGSMP's that are not EGSMPs. We will also characterize the subset of GSMPs that are not NRGSMPs.

In finding the steady state probability of the NRGSMP's, we will present an algorithm that transforms the NRGSMP into a semi-Markov process (SMP) with a bigger state space. The transformation preserves steady-state simulation, which allows us to determine the steady state probability of the NRGSMP from that of the SMP constructed. One of the drawbacks of this method is the *big* state space of the SMP created. In fact, the largeness of the state space is one of the main obstacles that the modeler faces when analyzing a stochastic model, it is known as *state space explosion*.

On the level of stochastic processes, several approaches have been introduced to deal with the issue of state space explosion. Several techniques could be applied to overcome this problem, such as state lumping, approximation methods by state truncation (aggregation), or bounding methods [9],[62],[78]. All these methods simplify the process on the state level by reducing the number of states. Other methods work by exploring properties of the model such as equivalence and partition, then reducing the state space [62],[78], but they are limited in applicability as the model needs to have certain properties. In this thesis, to overcome the size of the SMP created, we will introduce two algorithms:

- The first algorithm deletes certain states of the original GSMP while preserving the distribution of time to travel between non-deleted states. As will be seen in Chapter 5, the technique is limited in applicability to very particular states. The *passage of time equivalence* is not new, it was introduced by Bradley in [14] in the context of SMPs. So we use the same definition and extend it in the context of GSMPs**.** As far as we are aware, no similar simplification exists on the level of GSMP's

- The second algorithm deletes certain states of the resulting semi-Markov process, while preserving the mean passage-time between non-deleted states. As far as we are aware, the only existing simplification in the context of semi-Markov processes was introduced by Bradley in 2002 [14]. In his paper, Bradley introduced a simplification technique that preserves the exact passage-time distributions between pairs of non-deleted states; this in stochastic terms is a very strong equivalence, the two models under comparison should have strong similarity. In this chapter, the equivalence is less restrictive; processes would still be equivalent if they have the same average of passage time distribution between states rather than exact distributions; and the simplification procedure requires less time as its steps are straightforward. This equivalence is useful when the user is only interested in mean passage-time delays and not actual distributions. The *"mean-passage time equivalence"* preserves all performance measures that depend on mean passage time such as reliability and availability. So if we are interested in such measures, the simplification technique would help us reduce the size of the SMP and hence the complexity for the performance evaluation procedure.

1.3. Outline of the thesis

In the next two chapters, some background information will be provided: Chapter 2 introduces the different performance models and the advantages and disadvantages of modeling with each of them, and Chapter 3 introduces the two main existing methods for the numerical analysis of generalized semi-Markov processes: the method of supplementary variables and the method of embedded regenerative process. These methods provide a solution for a subset of GSMPs, the retriction being that at most one non-exponentially distributed clock can be enabled at any given time. Aside from being a severe restriction, the main problem with these methods is that it is hard to check whether a GSMP is part of this subset using static analysis. As a result, more restrictive conditions are set on the subset of solvable GSMPs to make the subset checkable using static analysis.

Chapters 4 to 6 introduce the results of this thesis: Chapter 4 introduces a new technique for the numerical analysis of generalized semi-Markov processes and compares it with previous methods. The new technique transforms a GSMP into an SMP, and calculates the steady state probabilities of the GSMP from that of the SMP created. This new method provides a solution for the steady state probability of a wider class of GSMPs, moreover, it is easy to check whether a GSMP is a subset of this class or not as will be dicussed in Chapter 7.

Chapter 5 introduces a method to remove states from GSMPs while preserving the distribution of time needed to travel between non-deleted states and the transient and steady state probabilities for a subset of the states of the automata is preserved as well. Chapter 6 deals with the issue of state space explosion of the SMP created; it deals with the problem by introducing a new simplification technique for semi-Markov processes. The technique deletes states from the SMP while preserving the average time to travel between non-deleted states, or what we call average delay simplification equivalence, the equivalence is shown to preserve measures that depend on the mean time to travel between states, examples of such measures are mean time to failure and mean time to repair. The technique also preserves the SSPs of a subset of the states of the automata. The whole picture becomes clearer with an application and a case study presented in Chapter 7. And finally, Chapter 8 provides our conclusions and suggestions for future research directions.

Chapter 2: Performance modeling

2.1. Introduction

Performance, dependability and performability techniques provide a method to study the behavior of computer and communication systems. Performance refers to the response time as seen by the users. Responsiveness determines a system's effectiveness and as a consequence affects the productivity of the users [7],[76]. Dependability modeling covers failure and repair related aspects of system behavior. Performance and dependability techniques are vital to most hardware and software systems. Software systems that perform customer service functions, such as ticket reservation systems and ATM banking systems must provide rapid responses to satisfy the customers. Hard real-time systems, such as flight control systems, must meet their response-time requirements to prevent disasters. If an automated flight-control system does not provide a rapid response, the airplane depending on it could crash.

Performance and dependability analysis can be studied separately, but sometimes, a measure that takes into account their interactions and trade-offs, or what is known as performability analysis, is needed. In fact, fault-tolerant systems are designed to guarantee continuity of service even in the presence of component failure [82],[47]. However, the performance of the system will be reduced in the presence of failure. For example, a system may operate as long as one of two components is operational, however, the system is assumed to deliver a higher performance when both components are operational. Performability analysis aims to capture the performance of the system in the presence and absence of failure and the interaction between the failure-repair behavior. For a survey of techniques and tools that can be used in reliability and performability analysis refer to [82].

To ensure that a system meets performance and reliability goals, performance has become an essential part of the software development process [77],[83].

2.2. Models

To evaluate the system or component during the development process for their performance, dependability and performability, a software designer has several options: "make an educated guess based on his past experience, build prototypes and make measurements, use discrete event simulation to model the system, or construct analytical models of the system" [47].

Assessing a *prototype* is not always possible during the implementation phase. Moreover it might not be possible to assure whether a prototype meets a performance, dependability or performability criteria, for example a system with high reliability might take months before it fails [72].

Discrete event simulation (DES) is commonly used in practice. Many software tools are available that could help in the construction and execution of DES models. However, simulation models are generally expensive to define because this involves writing and debugging a complex computer program. Moreover, they can be expensive to parameterize, because a highly detailed model typically requires a large number of parameters. And finally they are expensive to evaluate because running a simulation requires substantial computational resources, especially if narrow confidence intervals are desired [53],[47].

Analytical modeling is a cost effective alternatives for DES. They are the main focus of our research. Analytical modeling is based on constructing a model and analyzing it. A model is an abstract representation of the system; it is used to capture the essential characteristics of the system so that its performance can be reproduced; it should include sufficient information to make us understand the actual system's behavior [54], and such characteristics could be fault-tolerance, contention for resources, concurrency and synchronization…. A software designer has a wide range of different types of models to choose from. These models could be divided into two main categories:

a) *Queuing models*, such as product form queuing networks [48], can represent contention for resources. However they can not model failure, synchronization or concurrency.

b) *Stochastic models*, such as stochastic Processes [19],[62],[78], stochastic Petri-nets, (SPN), [24],[23],[60] and stochastic process algebras, (SPA), [46],[50],[54], can model interactions between system components. They are also known as state space models [47].

The stochastic processes that are mostly studied in the literature for performance and dependability purposes are, in increasing order of expressiveness, continuous-time Markov chains (CTMC), semi-Markov processes (SMP), and generalized semi-Markov processes (GSMP). The difference between them lies in the set of instants in the process life that satisfy the *Markov property*. A stochastic process satisfies the Markov property at the current time instant if the conditional probability distribution of the future states of the process, given the present state and all past states, depends only upon the present state and not on any past states; for more details refer to [19].

For CTMCs the Markov property holds at every instant of the process life, for SMPs the property holds at the instants of state change only, and for GSMPs, the property may never hold.

Because of the holding of the Markovian property at all instants of the process life, CTMCs are shown to only represent activities with an exponentially distributed duration [19]. The only candidates for representing systems with generally distributed activity durations are SMPs and GSMPs.

Although the state space models provide flexibility for modeling dependability, performance and performability, the state space of these models can be very large; in fact their state space grows much faster than the increase in system components. To deal with this issue, many high level specification techniques, such as stochastic Petri nets, and stochastic process algebras were introduced. These are higher level representations of stochastic processes. When the underlying stochastic process is a CTMC, the CTMC can be obtained automatically from these higher level models, moreover, effective methods for its solution are available [47].

In the remaining of this chapter, we will introduce CTMCs, SMPs and GSMPs; the information is taken from [53] and more details could be found in [12],[76]. Examples will be provided for all the state-based models. We will then introduce queuing models, Petri nets and process algebras; more details could be found in [12],[29],[54],[60],[23].

2.3. Stochastic Models

As discussed earlier, the state-based model, such as CTMCs, SPNs, and SPAs are all based on the notion of stochastic processes, so, in this section, we will present the stochastic processes that are mostly used in the context of performance modeling, and these are CTMCs, SMPs, and GSMPs. We start first by presenting some notions that are common to these three types of processes; for that purpose, any stochastic process that is a CTMC, an SMP or a GSMP will be referred to as a Markov-Like-Stochastic Process or MLSP.

An MLSP *X* is a state automata whose transitions are triggered by the occurrence of stochastically timed events associated with the occupied state. We denote by *X(t)* the state of the process at time *t*. Only continuous time models will be considered, in other words *t* takes its values from the set of positive real numbers. An MLSP is said to be *irreducible* if all states can be reached from all other states, by following the transitions of the process. An irreducible MLSP is also known as *strongly connected*.

Let X be an MLSP with finite state space $S=\{1,2,\ldots,n\}$ and let j be the starting state, in other words, $X(0)=j$, let $P_{ij}(t) = P\{X(t) = i | X(0)=j\}$ denote the probability of the process being in state i at time t given that the process was in j at time 0, then the row vector $P(t) = [P_{1j}(t), P_{2j}(t),\ldots,P_{nj}(t)]$ represents the transient state probability vector of the process. The steady-state probability (SSP) vector is $\pi=(\pi_1,\ldots, \pi_n)=lim_{t\to\infty}P(t)$. Note that $lim_{t\to\infty}P(t)$ may not exist, in which case the SSP would not exist, the conditions under which the SSP of the process exists will be presented later in this chapter.

In addition to transient state probabilities, "cumulative probabilities" can be useful sometimes. These are denoted by L and are given by

$$L(t)=\int_0^t P(u)du;$$

$L(t)=\{L_1(t),\ldots,L_n(t)\}$, *where* $L_i(t)$ denotes the "expected total time the process spends in state i during the interval $[0,t)$".

With these definitions, many interesting performance dependability and performability measures can be defined by assigning rewards to states or to transitions between states of the process to form what is known as a reward model (RM). In this section we consider state-based rewards only, the results are taken from [47],[54]. Let r_i be the reward rate assigned to state i. Then, the random variable $Z(t)=r_{X(t)}$ is "the instantaneous reward rate of the RM at time t". The reward that is accumulated over the interval $[0,t)$ is given by

$$Y(t)=\int_0^t Z(u)du)=\int_0^t r_{X(u)}du$$

Various measures can be defined from the random variables: $X(t)$, $Z(t)$, and $Y(t)$. A useful example is "the expected instantaneous reward rate" which is defined as follows:

$$E[Z(t)]=\Sigma_{i\in S}r_iP_{ij}(t)$$

And "the expected reward rate in steady state" is:

$$E[Z(\infty)]=\Sigma_{i\in S}r_i\pi_i$$

And “the expected accumulated reward” is:

$$E[Y(t)]=\Sigma_{i\in S}r_iL_i(t)$$

Some models can have states that do not have any outgoing transitions, these states are called absorbing states. In these models, the system would be in one of these absorbing state at equilibrium [68], and the limit as $t\rightarrow\infty$ of the expected accumulated reward is called the expected accumulated reward until absorption

$$E[Y(\infty)]=\Sigma_{i\in S}r_iL_i(\infty)$$

Given the RM framework the next question is: what are the appropriate reward rate assignments?” We will answer this question in the context of dependability analysis with two target measures: availability and reliability. For information on reward assignment for performability measures, the reader is referred to [47],[69].

Availability: availability measures the probability of a system going to an undesirable state, such as failure or service interruption. Availability is used for systems that tolerate interruption in service. In such systems failure is usually recoverable.

The simplest and most used availability measure is “the steady state availability”: It describes the probability of the system being in one of the desirable states at steady state. The availability is obtained by assigning a reward rate 1 to the desirable states and a reward rate 0 to down states

$$A=\sum_{s\in S} r_s\pi_s.$$

The mean time to failure or MTTF is another availability measure, it is related to A as follows:

$$A=MTTF/(MTTF+MTTR)$$

where MTTR is the average time spent in the fail state.

Reliability: reliability is the measure of uninterrupted service over a period of time. Reliability is used for systems that do not tolerate down times, such as flight control systems.

The simplest and most used reliability measure is also the mean time to failure, however, the failure here is unrecoverable, and so MTTF is the same as mean time to absorption (or MTTA). MTTF is again obtained by giving a reward rate of 1 to the up states and reward rate 0 to the down states. MTTF would then be given by

$$E[Y(\infty)]=\Sigma_{i\in S}r_iL_i(\infty)$$

So from the above we conclude that to obtain a complete description of a Markov-like-stochastic process, we need to find the transient state probabilities of the MLSP. However, it is often difficult to obtain such solutions [68], Moreover, in many practical situations one needs to know the behaviour of the system in steady state, in other words, one needs to know the behavior of the system when it reaches an equilibrium state, a state it reaches after being in operation for a sufficiently long time [68]. For that reason, it is necessary to know the conditions for the existence of these probabilities. This will be discussed across this section. When such limit exists, the system is said to reach equilibrium or steady state, and then problem translates to finding the steady state probabilities. The calculation of TSPs or SSPs is referred to as finding *the solution* for the process.

In the remainder of this section, we will present the different types of Markov-like-stochastic processes; we start first by presenting the exponential model: continuous-time-Markov chains.

2.3.1. Exponential Models

Models with exponentially distributed holding times have been extensively studied [2],[25],[54],[60]. In this sub-section, we will present CTMCs and their properties, we start first by defining a poisson process.

2.3.1.1. Continuous-time Markov chains

A Poisson process is a counting process in which interarrival time of successive jumps are independently and identically distributed exponential random variables. For more details refer to [19].

A CTMC is an MLSP; each state in the automaton is associated with several Poisson processes. A transition between any two states of the automaton is governed by the jump of one of the Poisson processes associated with the occupied state. The different Poisson processes associated with a state compete to trigger the next transition. The distribution of the time spent in a state of a CTMC, or what is known as the soujourn time in the state, is shown to be exponentially distributed [19].

Formally, a Continuous time Markov chains *X* (CTMC) is made up of the tuple (S, s_0, q) where

1. S is a nonempty set of states,
2. $s_0 \in S$ is the starting state,
3. $q : S \times S \rightarrow \Re$ (where $\Re$ is the set of real numbers). For $i \neq j \in S$, q_{ij} is called the instantaneous transition rate from state i to state j. It is the parameter of the exponential distribution of the sojourn time in state i, given that the next state to be visited is j, in other words, it is the parameter of the distribution of the interarrival time of the Poisson process associated with the transition from i to j. q_i is called the exit rate for state i, and it is the parameter of the exponential distribution of the sojourn time in state i. In other words: $q_i = \sum_{i \neq j \in S} q_{ij}$, and $q_{ii} = -q_i$. For more details refer to [19]. The matrix $[q_{ij}]$ is denoted by Q.

Continuous-time Markov chains with small state space are commonly represented as a state transition diagram. Each state is represented by a node. Possible transitions between the different states are represented through arcs. The arcs are labelled by the parameters of the exponential distributions governing the transitions: q_{ij}.

For more details and for examples of CTMCs refer to [12],[19],[25],[68],[78].

2.3.1.2. Deriving the steady state probabilities for continuous-time Markov models

Let X be a CTMC. Recall that $P_{ij}(t) = P\{X(t) = i| X(0)=j\}$ is the transient state probability for state i, and that $P(t) = [P_{1j}(t), P_{2j}(t),\ldots,P_{nj}(t)]$ represents the transient state probability vector of the CTMC. Then

- The transient behavior of the CTMC can be described by the Kolmogorov differential equation [19]: $dP(t)/dt=P(t)Q$ given $P(0)$, where $P(0)$ represents the initial probability vector (at time $t = 0$).
- The steady-state probability vector $\pi=(\pi_1,\ldots, \pi_n)=\lim_{t\to\infty}P(t)$ satisfies: $\pi Q=0$ *and* $\Sigma_n\pi_i=1$ [19].

Theorem 2.1. A steady state probability distribution exists for every finite and strongly connected continuous-time Markov Chain (irreducible). [68]

∎

The different methods for steady state calculation are:

1. The direct methods which are numerical methods that compute solutions to mathematical problems in a fixed number of operations [78].

2. The iterative methods, which are the mostly used methods, begin from some initial approximation and produce a sequence of intermediate results, which are expected to eventually converge to the solution of the problem [78].

3. Other methods include: projection Methods, decompositional methods …[78].

The complexity of the methods above lies between $O(2n^{2.7})$, and $O(n^3)$. In fact it is $O(\frac{n^3}{3})$ in most of the methods.

2.3.2. Non Exponential Models

As explained in the previous section, because of their memoryless properties, CTMCs can be easily analyzed through straightforward application of numerical analysis. And that is the reason behind the popularity of the exponential distribution. However, the exponential assumption is not always realistic [34],[25],[41]. The following list, taken from [61], contains few examples of events that can not be modeled using exponential distributions.

- If only the minimum and maximum of some quantity is known and more information is not available, the uniform distribution would be a good choice.
- File transmission times in the internet and file sizes on a host give evidence of heavy-tail distributions.
- The Weibull distribution is common in reliability, since it has an age-dependent failure rate.
- Clock cycles in computers are fixed, i.e. they have deterministic distributions.
- Repair times and scheduled maintenance intervals have often a fixed length.

So the above list proves the need for models with non-exponential distributions, in fact the focus on non-exponential distributions has flourished in the past 20 years in the area of SPN [43],[23],[55],[63],[64] and SPA [13],[18],[33],[51],[79]. If we allow transitions to be delayed by non-exponential distributions, the Markov property need not hold anymore, the future behavior of the process might depend on a distribution that was started in the past and has not triggered a transition yet. A stochastic process that allows non-exponential transitions and has the Markov property at the time of state change is the semi-Markov process. In a semi-Markov process, all distributions that govern transitions are initialized every time we enter a state; as a result, we do not need to memorize the lifespan of distributions from past states.

In the next sub-section, we will informally present GSMPs and SMPs. Formal definitions and illustrations will then follow.

2.3.2.1. Semi-Markov processes and generalized semi-Markov processes

A generalized semi-Markov process (GSMP) is a state automaton whose transitions are triggered by the occurrence of stochastically timed events associated with the occupied state. A set of active events $A(s)$ is associated with each state s of the automaton, these events compete to trigger the next transition. Each of these events has its own distribution for determining the next state. At each transition to a state s, a set $K(s)$ of new events will be scheduled. For each of these new events, a clock indicating the time when the event is scheduled to occur is set (according to the random distribution associated with the event). If an event *e'* occurs causing a transition from state s to state s', and if another event *e* was active in state s, then *e* is either associated with the next state (i.e. $e \in A(s')$), and its clock continues to run; or *e* is not associated with the next state s', in that case, it is abandoned (or we say aborted), i.e. its associated lifetime is discarded and it is considered inactive. A transition between two states is labeled by an event *e* and a set of events *E* that are aborted, written $s \xrightarrow{e}_E s'$. This means that if the clock of event *e* expires (or we say simply if event *e* occurs) in s then the process aborts the events in *E* and moves to state s'. Only one transition out of s should be labeled with a given event *e* (determinism). The active events in state s' would then be $A(s') = A(s) - (e \cup E) + K(s')$. The events in the set $A(s) - (e \cup E)$

keep their residual lifetime, while events in $K(s')$ are assigned a lifetime according to their distributions (initialized). Note that we need not represent the set of aborted events *E* because they can be deduced from the functions A and K.

SMPs are GSMPs with the property that *K(s)=A(s)* for all *s*. In other words, all active events in a state are initialized once we reach the state. Because of this restriction, SMPs satisfy the Markov property at the time of state change.

A. SMP definition.

Definition 2.1: Semi-Markov processes.

A Semi-Markov Process (SMP) [25],[68] G is made up of the tuple $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$ where

- S is a nonempty finite set of states including the initial state s_0,
- E is the set of events,
- $K : S \to \wp_{fin}(\mathrm{E})$ is the event setting function which represents all the events that are initialized when we reach a state (note that these are the only active events in the state).
- For every $s \in S$, the function $F_s : K(s) \to (\Re \to [0,1])$ assigns the event distribution functions such that for all e in $K(s)$, $F_s(e)(x) = 0$ for $x < 0$ and $\lim_{x\to\infty} F_s(e)(x) = 1$. (As mentioned before, the distribution of an event depends on the state it was initialized in)
- $\mapsto \subseteq S \times \mathrm{E} \times S$ is the set of edges where $s \overset{e}{\mapsto} s'$ means that if event *e* occurs first in s then the process moves to state s'. Note that the process is deterministic, in other words, if $(s, e, s_1) \in \mapsto$ and $(s, e, s_2) \in \mapsto$ then $s_1 = s_2$.

An SMP with finite states can be represented as a labeled transition system (refer to Figure 2). Every state *s* is annotated with a set of events inside brackets, these are the events that are active in that state ($K(s)$). Transitions out of a state are annotated with an event, say *e*, meaning that the transition takes place when event *e* occurs, the distribution governing the transition is given by $F_s(e)(x)$. The numbers inside the state are labels that identify the state.

Let $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$ be an SMP, let $s, s' \in S$ such that $s \xrightarrow{e} s'$, then:

- The conditional probability of moving out of state s to state s', given that the process is currently in state s, is calculated as follows: $p_{ss'} = \int_0^\infty \frac{dF_s(e)(x)}{dx} \prod_{f \in K(s)} (1 - F_s(f)(x))dx$, i.e. it is the probability that event e occurs first from state s. For the different states in S, these probabilities form a Matrix $[p_{ss'}]$ referred to as the embedded Markov chain.

- Let S_s be the set of all states that are directly accessible from s, in other words, S_s is the set of all states { $r \in S$, such that $s \xrightarrow{e_r} r$ }. Then, the mean waiting time in state s, M_s, is calculated as follows: $M_s = \sum_{r \in S_s} p_{sr} E(T_{sr})$ where T_{sr} is the conditional waiting time in state s given that the next state to be visited is r, and $E(T_{sr})$ is the expected value of T_{sr}.

For more information on the above definitions and derivations, the reader is referred to [68].

B. Embedded Markov chain

In this sub-section, we will briefly describe the embedded Markov chain, for more background, the reader is referred to [25],[59],[68].

Let $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$ be an SMP, let $s, s' \in S$ such that $s \xrightarrow{e} s'$. As pointed out in the previous sub-section, the conditional probability of going from state s to state s':

$$p_{ss'} = \int_0^\infty \frac{dF_s(e)(x)}{dx} \prod_{f \in K(s)} (1 - F_s(f)(x))dx.$$

For the different states in S, $[p_{ss'}]$ form a matrix for the embedded discrete-time Markov chain, or simply the embedded Markov chain. The embedded Markov chain describes the probability of moving between the states of the process without regard to the sojourn time in the different states of the process. In other words, it refers to the state of the process at the *n+1* transition given its state at the *n*th transition; regardless of the time of occurrence of these transitions (we consider here the case where this probability is independent of *n*). The probability $p_{ss'}$ is also denoted by $P(s'(n+1) \mid s(n)) = P(s'(1) \mid s(0))$. Now, the conditional probability of moving to state s at the *n*th transition, given that the chain started from state s_0, is denoted by $P(s(n) \mid s_0(0))$.

For any $s \in S$, the steady state probability of state s, π_s, is probability of being in state s in steady state, i.e. $\pi_s = \lim_{n \to \infty} P(s(n) \mid s_0(0))$. The steady state probability of the embedded Markov chain is determined using the following equations:

$$\pi_s = \sum_{r \in S} \pi_r p_{rs} \text{ and } \sum_{s \in S} \pi_s = 1$$

Theorem 2.2. A steady state probability distribution exists for every finite, irreducible, and ergodic [68] discrete-time Markov chain.

C. Example of an SMP.

Example 2. The following example is taken from [25]: A system has two identical devices, each of the devices may fail independently of the other with constant failure rate *λ>0*, in other words, the time until a failure occurs is exponentially distributed with parameter λ. The failure is modeled by event f. State 1 represents the normal mode of operation, both units are working (see Figure 2). Upon failure of one device, the process moves to state 3, and the service is suspended for a random amount of time, during which the faulty device is identified, the identification of the faulty device is modeled by event I. The distribution $v(t)$ associated with event I according to which the system moves into the next state (state 2) is of general nature.

We assume that no failure can occur in State 3. Subsequently (in State 2), the working unit resumes service while the faulty one undergoes repair. The repair rate μ is constant, and repair is modeled by event R. State 4 stands for "both units down", and it can be entered or exited only through State 2. The failure rate in State 1 is 2λ because both devices are up, and similarly the repair rate in State 4 is 2μ (assuming independent failure times).

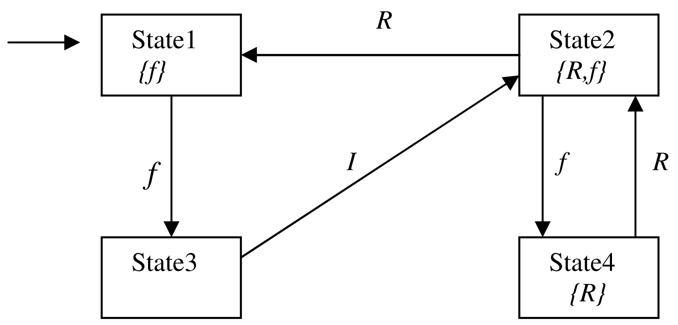


Figure 2. An example of a semi-Markov process

D. Transient and steady probabilities for semi-Markov models

Semi-Markov processes are heavily used in this thesis; the results of Chapter 4 rely on the transient state probability theory for SMPs. Hence, in this sub-section, we will outline the different methods for determining the transient and steady state probabilities for semi-Markov processes, and their complexities.

Let *{X(t), t >= 0}* be an SMP, and assume that, $X(0) = j$, recall that $P_{ij}(t) = P\{X(t) = i | X(0)=j\}$ is the transient state probability of the process for state *i*, and that $P(t) = [P_{1j}(t), P_{2j}(t),\ldots,P_{nj}(t)]$ represents the transient state probability. The steady state probability can be obtained from the formulae: $\pi=(\pi_1,\ldots, \pi_n)=lim_{t\to\infty}P(t)$. A different, but equivalent, definition for steady state probabilities was presented in [27]; the definition considers the proportion of time spent in every state of the SMP:

$$\pi_i = \lim_{t\to\infty} \frac{1}{t}\int_0^t P_{ij}(x)dx \qquad (1)$$

The different methods for determining the TSP of an arbitrary *n*-state SMP include the following:

1. Cox and Miller [27] derive a matrix technique for solving the TSPs of a two-state semi-Markov process. The technique extends directly to arbitrary number of states.

2. In Bradley [15], a different method is presented. It concentrates on finding the TSP of a two-state semi-Markov process. Then the method is generalized to cover an arbitrary state process. The generalization is achieved by reducing a general *n*-state process into two states using the process of stochastic aggregation;' for more details, refer to [14].

3. In Pyke [73] a different method is presented. It provides a formula for calculating the vector matrix *P(t)* from its Laplace transform. As in Bradley's method above, this method concentrates on finding the TSP for two-state processes; the result is then generalized in the same way as in [15] to an arbitrary state process.

In all the three methods, SSPs are derived from TSPs using formula (1) above. Another direct method to obtain the SSP is illustrated in the theorem below:

Theorem 2.3. Let $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$ be an SMP with finite state space. Let P be the transition matrix for the embedded Markov chain of G. Let $V = \{v_s\}_{s \in S}$ be the probability vector that is the solution of the equation $V = VP$. Then, if V exists, and if the mean waiting time in state s, $M_s < \infty$ for all $s \in S$, then the steady state probabilities exist and can be calculated as follows: $\pi_s = \dfrac{v_s M_s}{\sum_{r \in S} v_r M_r}$.

The complexity of the methods above is dominated by the inversion of an $n \times n$ matrix, where n is the number of states in the SMP. The inversion of an $n \times n$ matrix is possible through several different methods [84], the complexity of the procedure (finding the TSP or SSP) for the different methods is between $O(n^{2.376})$ and $O(n^3)$.

As discussed in the first Chapter, SMPs are not suitable for representing event concurrency. Hence the need for more general models such as generalized semi-Markov processes.

2.3.2.2. Generalized semi-Markov process

In the next sub-section, we formally define generalized semi-Markov processes and present their properties.

A. Formal definition and examples

Definition 2.2: Generalized semi-Markov processes.

A GSMP is a tuple $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ where:

- S is a nonempty set of states including the initial state s_0,
- E is the set of events,

- $K : S \to \wp_{fin}(\mathrm{E})$ is the event setting function which represents all the events that are initialized when we reach a state. Note that, because of their memoryless property, all active events in a state s that have an exponentially distributed lifetime are assumed to be initialized in s (i.e. they are assumed to belong to $K(s)$)

- For every $s \in S$, the function $F_s : K(s) \to (\mathfrak{R} \to [0,1])$ is an *s*-dependant function that assigns the event distribution functions such that for all e in $K(s)$, $F_s(e)(x) = 0$ for $x < 0$ and $\lim_{x\to\infty} F_s(e)(x) = 1$. (As mentioned before, the distribution of an event depends on the state it was initialized in).

- $A : S \to \wp_{fin}(\mathrm{E})$ is the set of events that are active in a state, note that $K(s) \subseteq A(s)$ for all $s \in S$.

- $\mapsto \subseteq S \times \mathrm{E} \times S$ is the set of edges where $s \overset{e}{\mapsto} s'$ means that if event *e* occurs first in s then the process moves to state s'. Note that the process is deterministic, in other words, if $(s, e, s_1) \in \mapsto$ and $(s, e, s_2) \in \mapsto$ then $s_1 = s_2$.

A GSMP is depicted as a labeled transition system, as shown in

Figure 3. Every state *s* is annotated with a set of events inside brackets, these are the events that are initialized in that state ($K(s)$). Transitions out of a state are annotated with an event, say *e*, meaning that the transition takes place when event *e* occurs. The numbers inside the state are labels that identify the state.

Example 3. We consider the model of a machine that receives requests and services them. The machine can service one request at a time, and requests are generated when the machine is not in service. The request generation is modeled by event r and servicing a request is modeled by event s, both have a generally distributed lifetime duration. The machine keeps working for a constant period of time then undergoes tune-up. If the machine is servicing a request when tune-up is due, the machine aborts the current service to undergo the tune-up. The interval between two consecutive tune-ups is constant and is modeled by event a. The tune-up process is generally distributed and is modeled by event u. While in service, the machine can fail, at that time it has to undergo repair, the failure and repair are modeled by events f and p, respectively, they both have generally distributed lifetimes. If the machine fails and is repaired, it reinitializes event a, Transition d is immediate. Note that a might have a different lifetime distributions in state 4 and 0, because the tune-up time after a repair is not so urgent. The model is depicted in Figure 3.

Now, we present another example of a GSMP:

Example 4 (refer to [61]): Consider a G/G/1 queue of size 3. The arrival and the service are represented by a and s respectively. The number inside a state represents the state label as well as the number of customers inside the system (number of customers waiting+ number of customers in service). Refer to Figure 4.

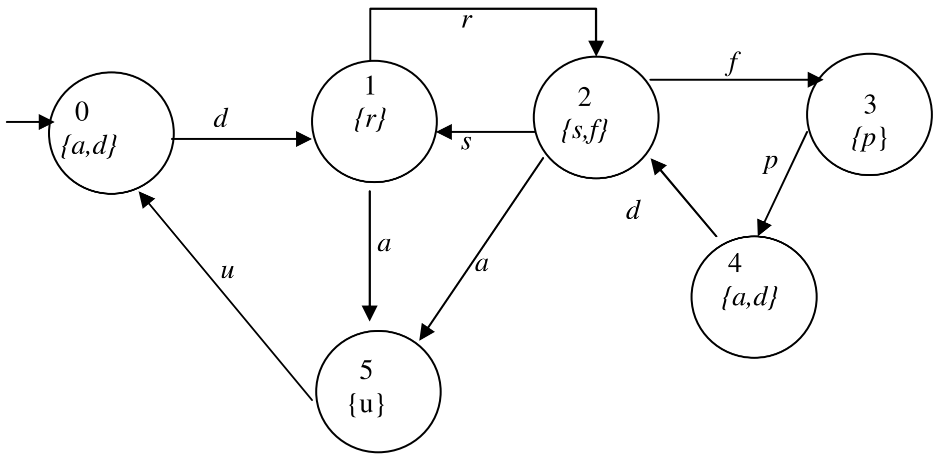


Figure 3. Example of a GSMP

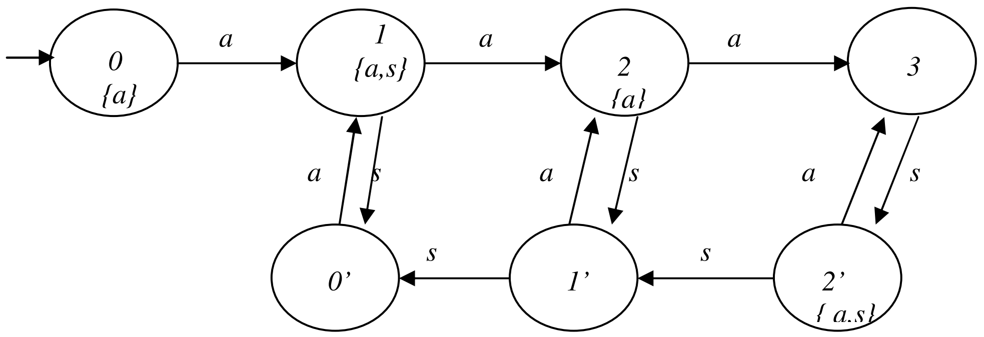


Figure 4. G/G/1/

B. Transient and steady state probabilities for GSMPs

Transient State Probabilities: as explained in Chapter 1, finding the TSPs for GSMPs is difficult because of the absence of the Markov property. In the next chapter, we will present the different methods available to analytically find the transient state probabilities of GSMPs.

Steady state Probabilities: once the TSPs are calculated, the SSPs are derived from TSPs using Formulae (1) in Section 2.3.2.1.C. However, for a subclass of GSMPs, referred to as insensitive GSMPs [67],[75], the steady state probability can be derived through a simple application of numerical analysis. Insensitivity results were originally presented by Matthes in [67].

Definition 2.3: Insensitive GSMPs.

A stochastic process is said to be *insensitive* if its steady state distribution depends only on the mean of the random variables representing residence time in the states of the process.

Matthes showed the following result [67]:

Theorem 2.4. Given a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$, the following two statements are equivalent:

1. The process is insensitive to the events of E. That is, the distributions of the lifetimes of the events of E may be replaced by any other distribution with the same mean.

2. When all events of E are assumed to be exponentially distributed, the flux (i.e. the instantaneous rate) out of each state due to the occurrence of an event of E is equal to the flux into that state due to the activation of that event.

∎

So if a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ is insensitive to the events in E, and if $G' = (S, s_0, \mathrm{E}, F', A, \mapsto, K)$ is the GSMP obtained from G by replacing the distributions associated with events in E with the exponential distributions having the same mean - in other words $F_s(e)$ and $F'_s(e)$ have the same mean for all $e \in E$, and $F'_s(e)$ are exponential distributions - then G and G' have the same steady state probabilities.

Conditions for insensitivity have been investigated by several researchers in the context of SPN. In [34], authors investigated the notion of insensitivity when the stochastic model underlying the SPN is an SMP. In [3],[49], the authors investigated insensitivity in the context of GSMPs, the restrictions presented identify a class of allowed general distributions along with some restrictions on the concurrently enabled transitions that result in an insensitive GSMP. Clark et al. translate these restrictions to the context of SPA [22].

2.4. Queuing Models

2.4.1. Introduction

Queues have been studied by mathematicians for more than 100 years [48], and the first applications they looked at were in telephone exchanges. Queues became popular with computer scientists about four decades ago, at which time, they came to the realisation that single queues, and networks of queues could be used as models to study the performance of computer systems. Recently, the growth in computer systems' models resulted in models that can not be expressed using queuing networks. However many people still view performance analysis as the synonym of queuing theory.

2.4.2. Single Queues

In single queues, customers arrive at a service facility where one or more servers are waiting to service these customers. Servers are usually assumed to be indistinguishable in terms of the service they can provide. If a customer cannot gain access to a server it must wait in a queue, until a server becomes ready. Upon completion of the service request, a customer departs from the facility, the next customer is then selected from the queue according to a predefined service discipline.

The following items are required in the study of queuing models:

Arrival Pattern of Customers: servicing of customers depends on the distribution function of the inter-arrival times. Inter-arrival times are usually assumed to be exponentially distributed, i.e. they correspond to a random arrival with a large customer population. However, this scenario need not be always true.

Service Time Distribution: is the time that a server spends servicing a customer. The distribution function of the service time is usually assumed to be exponential.

Number of Servers: If one server is available, the service facility can only serve one customer at a time; other customers have to wait in the queue until the server is available; the next customer is chosen depending on the service discipline. If infinitely many servers are available, then customers never wait for service and the queue is always empty. If a fixed number of servers (usually denoted c) are available, then arriving customers wait in the queue only if the number of customer in the facility exceeds c

Queue Capacity: Customers who cannot receive service wait in the queue for a server to become available. The number of customers waiting may grow, depending on the inter-arrival and service distributions. If the queue has a finite capacity c, then it may become full. In this case, any additional customer is turned away and lost.

Service Discipline: When several customers are waiting for service, a discipline for selecting the next customer must be provided. Some of the commonly used disciplines are:

- FCFS first come first serve (or FIFO first in first out).
- LCFS last come first serve (or LIFO last in first out).
- PRI priority. The assignment of priorities to customers according to the service they require

2.4.2.1. Solving single queues

If inter-arrival time and service time are exponential, then the queue can be modelled as a continuous-time Markov chain [53]. However, the SSP of the underlying continuous-time Markov chain can be obtained through simple application of linear algebra (it is a function of the arrival time, the service time and the size of the queue). Hence common performance measures can be obtained through simple application of linear algebra without referring to the underlying CTMC and without the need to solve it. Examples of such measures are: mean number of customers in the queue, mean service time and the probability of the system being idle.

If inter-arrival time and/ or service time are non-exponential, then the queue can be modelled as a generalized semi-Markov process. Solutions of these queues are available (without referring to the underlying GSMP) for the case where either the service time or the arrival time is non-exponential. For more details refer to [12],[25],[68].

2.4.3. Networks of Queues

If we view a computer as a set of devices, and customers (or requests) move from one device to the other sequentially, then we can model the system as a queuing network. Each device is represented by a separate queue or service centre. Customers in the network correspond to the users in the whole system. A customer may move from one service centre to another in the system, the pattern in which customers move around is predefined. Each service centre is a single queue, however, its characteristics (arrival, service time,…) are not independent of the other service centres within the network.

A queuing network is characterized by: the network topology, the characteristics of each service centre, and by its customers. If the network is closed, i.e. if external arrivals are not allowed, then the number of customers inside the network is fixed an must be known. If the network is open, then the arrival process to each service centre is needed.

A queuing network is represented as a directed graph, nodes represent service centres, arcs represent the paths customers can take when moving between service centers. The state of the network is defined by the number of customers in each service centre.

The analysis of a network of queues is based on the analysis of the underlying stochastic process. The underlying stochastic process is usually a CTMC. The state of the process includes the number of customers in each service center [6].

For a closed network, the state of the underlying process grows exponentially relative to the number of service centers and the number of customers in the network, for an open network, the number of states in the underlying process is infinite. For that reason, extracting performance measures from a network of queues is not always possible. However, for a subclass of queuing networks, referred to as product form queuing networks straightforward means of extracting performance measures have been found [6].

2.4.5. Product Form Queuing Networks

The term “product form” represents the fact that the steady state probability of the queuing network can be derived as the product of the steady state distributions of each of the service centres that make up the network. In other words, once the different service centres reach equilibrium they behave independently of each other; for more information refer to [48],[53].

Recall that the state of a single service facility can be characterized by the number of customers currently in the system. In a queuing network the state of the system is characterized by the number of customers waiting at each of the service centres. This is usually represented as a tuple.

Product form solution for a network of queues holds under certain assumptions. These assumptions are defined on the Markov process underlying the network. The precise characterization of product form queuing networks is not easy, for that reason, conditions that are sufficient to ensure product form have been derived; an example is the *quasi-reversibility of every service centre*: quasi-reversibility states that the current state of a service centre, the past departures and the future arrivals are independent. For more information, refer to [6].

Under the assumption of a product form queuing network, using the already established formula for individual queues, the steady state probability of the network can be obtained without the need to develop the underlying process.

2.4.5. Expressiveness

Some aspects of computer and communication systems can not be represented by queuing networks. Some of these aspects are listed below. Ways for providing solutions for systems with such aspects remain topics for research.

Simultaneous resource possession: In a computer system a job may be using more than one resource in the system simultaneously. A solution for this problem is to use Layered Queuing Networks [85].

Bulk arrivals: the arrival rate between customers is not always independent, examples are bulk arrivals (i.e. arrivals that occur in batches).

2.4.6. Queuing Networks: Pros and Cons

Queuing models can be constructed, and evaluated relatively easily [12]. The behavior of each service is expressed based on the six characteristics of Kendall's notation [12]. However, the expressivity of queuing networks is limited. Queuing networks cannot represent systems in which more than one resource must be simultaneously retained, or systems in which there is internal concurrency. Some work has been done to remedy these cases, but its applicability is still limited [53],[85].

2.5. Petri Nets

In this section, we will briefly present the notion of Petri nets; for more information and for examples refer to [23],[60].

2.5.1. General Notions

Petri nets provide a graphical notation for the formal description of the dynamic behaviour of systems. They are particularly well suited to systems which exhibit concurrency, synchronization, mutual exclusion and conflict. The primitives of the notation are the following:

- PLACES are used to represent conditions or local system states, e.g. a place may relate to one phase in the behaviour of a particular component.
- TRANSITIONS are used to describe events that occur in the system; these will usually result in a modification to the system state. The occurrence of the event in the system is represented by the firing of the corresponding transition in the Petri net.
- TOKENS are identity-less markers that reside in places. The presence of a token in a place indicates that the corresponding condition or local state holds.
- ARCS specify the relationships between local states or conditions (places) and events (transitions). An arc from a place to a transition is termed an input arc. This indicates the local state in which the event can occur. An arc to a place from a transition is termed an output arc. This indicates the local transformations which will be induced by the event. Tokens move between places according to the firing rules imposed by the transitions.

A transition can fire when each of the places connected to it has at least one token; when it fires, the transition removes a token from each of these places and deposits a token in each of the places it is connected to by output arcs. This is called the firing rule. Sometimes a transition will require an input place to contain two or more tokens before it can fire. In this case, rather than draw more than one arc between the place and the transition, we denote the multiplicity of the arc by a small number written next to the arc. Similarly for output arcs. The state of the system combines information about all the local states. Since each local state is represented by the number of tokens present in a particular place, the state of the system is represented by a tuple, with one entry for each place, and the value of the entries denoting the number of tokens in that place. This is termed a marking of the net.

A Petri net consisting of places and transitions linked by arcs is incomplete if it does not also have tokens in some places. The initial placing of tokens is called the initial marking, this represents the starting state of the system.

Starting from an initial marking and following the firing rule we can progress through the states of the model. Continuing in this way, recording all the states we see and stopping only when we can reach no states that we have not already seen, we obtain all the possible states of the model. This is called the reachability set; it is the set of all possible markings that a net may exhibit, starting from the initial marking and following the firing rules. Different initial markings might lead to different reachability sets. This is why the initial marking is an important part of the model definition. If we record all possible states and all possible transitions between those states, we obtain the reachability graph. This is a graph in which the nodes are the reachable markings and the arcs between nodes represent the possible transition firings which may move the model from one marking to the other.

If we wish to extract timing information from a model we must represent timing information about the system in the model when it is constructed. In the case of Petri nets there has been a variety of suggestions of how to introduce timing information into Petri net notation.

If we consider the reachability graph of a Petri net model it resembles the state transition diagram of a Markov-like process. Stochastic Petri Nets (SPN) formalise this intuitive correspondence. Given a Petri net model (complete with initial marking): we associate a state in the Markov process with every marking in the reachability graph of the Petri net; we associate an event, or transition, in the Markov process with each firing of a transition in the Petri net which causes the corresponding change of marking. Since an exponentially distributed delay is associated with each event in a Markov process, and transitions in the Petri net correspond to events, in an SPN model an exponentially distributed delay is associated with each transition in the net structure. Thus each transition in an SPN has a firing rate which is the parameter of the corresponding exponential distribution, and transitions are sometimes termed timed transitions.

Non-exponentially distributed delays are added to a PN in the same way, the resulting process is referred to as non-Markovian Petri net, for more information refer to [34],[38],[23].

One of the advantages of SPN models is the straightforward correspondence between the reachability graph of the SPN and the state transition diagram of the Markov-like process it generates.

2.5.2. Petri Nets: Pros and Cons

- The time required to model construction is often greatly reduced compared with Markov-like processes. However some additional skill is required to learn the notation and semantics of the nets [53].
- Solution of the Petri net is obtained by generating the underlying Markov-like Process (the reachability graph of the Petri net is generated and then the Markov-like process). In the case of SPN, deriving the Markov process is a straightforward task. Once the Markov Process is generated, the solution proceeds numerically. So deriving performance is handled the same way as in the Markov process case. However the identification of states of interest could be easier using the Petri net.
- State space explosion and problem size are the major problem in steady state distribution of the underlying process. Extensive research has been dedicated to finding efficient approaches to dealing with the problem [9].

2.6. Process Algebras

In this section we consider another class of performance modelling paradigms:- stochastic extensions of process algebras. Like queuing networks and stochastic Petri nets, these formal languages can be regarded as high level model specification languages for low-level stochastic models. As we will see, the development of stochastic process algebras, or SPA, has been very similar to that of SPN: in both cases an untimed formalism, used for studying the correct functional behaviour of systems, is extended by associating generally distributed delays with actions and reachability analysis is used to construct a corresponding stochastic process. The advantages of SPAs are that they incorporate the attractive features of process algebras and thus bring to the area of performance modelling several attributes which are not offered by the existing formalisms. Perhaps the most important feature is the compositionality which is inherent in the models and can be exploited during their analysis. Several stochastic process algebras have appeared in the literature but they are all broadly similar. Here we will concentrate on PEPA (Performance Evaluation Process Algebra [54]).

Process algebras are abstract languages used for the specification and design of concurrent systems. The most widely used process algebras are Milner's Calculus of Communicating Systems (CCS) and Hoare's Communicating Sequential Processes (CSP) and the SPAs take inspiration from both these formalisms. Models in CCS and CSP have been used extensively to establish the correct behaviour of complex systems by deriving qualitative properties such as freedom from deadlock or livelock.

In the process algebra approach, systems are modelled as collections of entities, called agents, which execute atomic actions. These actions are the building blocks of the language and they are used to describe sequential behaviours which may run concurrently, and synchronizations or communications between them.

In CCS two agents communicate when one performs an action, *a* say, while the other performs the complementary action *a*. The resulting communication action is regarded as an internal action that is invisible to the environment. Agents may proceed with their internal actions simultaneously but it is important to note that the semantics given to the language imposes an interleaving on such concurrent behaviour, i.e. it is not possible for two actions to occur simultaneously. The grammar of the language makes it possible to construct an agent which has a designated first action (prefix); has a choice over alternatives (choice); or has concurrent possibilities (composition).

The communication mechanism is different in CSP as there is no notion of complementary actions: this is a major distinction between CCS and CSP. In CSP two agents communicate by simultaneously executing actions with the same label. Since during the communication the joint action remains visible to the environment, it can be reused by other concurrent processes so that more than two processes can be involved in the communication. This is the communication mechanism adopted in the SPA languages and in Lotos.

2.6.1. PEPA

According to Hillston [53], "Process algebras offer several attractive features which are not necessarily available in existing performance modelling paradigms. The most important of these are compositionality, the ability to model a system as the interaction of its subsystems, formality, giving a precise meaning to all terms in the language, and abstraction, the ability to build up complex models from detailed components but disregarding internal behaviour when it is appropriate to do so. Queuing networks offer compositionality but not formality; SPN and GSPN offer formality but not compositionality; neither queuing networks nor Petri nets offer abstraction mechanisms".

PEPA extends classical process algebra by associating a random variable, representing duration, with every action. These random variables are assumed to be exponentially distributed and this leads to a clear relationship between the process algebra model and a Markov process.

PEPA models are described as interactions of components. Each component can perform a set of actions: an action a is described as a pair (e,r), where e is the type of the action and r

is the parameter of the exponential distribution governing its duration. Whenever a process P can perform an action, an instance of a given probability distribution is sampled: the resulting number specifies how long it will take to complete the action. A small set of constructors is used to build up complex behaviour from simpler behaviour. The constructors are: sequential composition (prefix), choice, synchronization (cooperation) and abstraction (hiding). We explain each of them below, in terms of a extremely simple model of a web based information system.

- Prefix (.): A component may have purely sequential behaviour, repeatedly undertaking one activity after another and eventually returning to the beginning of its behaviour.

- Choice (+): A choice between two possible behaviours is represented as the sum of the possibilities. A race condition is assumed to govern the behaviour of simultaneously enabled actions and the continuous nature of the probability distributions ensures that the actions cannot occur simultaneously. Thus a sum will behave as either one summand or the other. When an action has more than one possible outcome, e.g. the display action in the browser, it is represented by a choice of separate actions, one for each possible outcome. The rates of these actions are chosen to reflect their relative probabilities (decomposition principle).

  Note that in a GSPN we would represent this situation by a single timed transition to represent the display action, which when it fired enabled two immediate transitions with weights p_1 and p_2 to reflect the different possible outcomes.

- Cooperation $P \infty_L Q$: The cooperation constructor represents a parallel composition between P and Q for all the actions not in the set L. Actions in this set L require the simultaneous involvement of both components. The resulting action, a shared action, will have the same type as the two contributory actions and a rate reflecting the rate of the action in the slowest participating component. Note that this means that the rate of a passive action will become the rate of the action it cooperates with.

- Abstraction (/): It is often convenient to hide some actions, making them private to the component or components involved. The duration of the actions is unaffected, but their type becomes hidden, appearing instead as the unknown type τ.

The formality of the process algebra approach allows us to assign a precise meaning to every language expression. This implies that once we have a language description of a given system its behavior can be deduced automatically. The meaning, or semantics, of a PEPA expression is provided by a formal semantics, in the structured operational style, which associates a labeled transition system with every expression in the language. This form of directed graph shows the possible evolutions of the model.

2.6.2. Non-Markovian Process Algebras

The algebra we discussed above have the property that delays are governed by exponential distributions. In this section we will briefly discuss how arbitrary, nonMarkovian probability distributions can be represented in process algebras. The information is taken from [61].

Because of the memoryless property of the exponential distribution, the parallel composition of two exponentially distributed events can be incorporated into an interleaved setting, i.e.

$$a \infty_{\phi} \mathrm{b} = a.\mathrm{b} + \mathrm{b}.a$$

If we allow actions to be delayed by arbitrary distributions, the above law becomes invalid, in other words, if an action *a* has arbitrary distribution *F* and an action *b* has arbitrary distribution *G* then:

$$a \infty_{\phi} \mathrm{b} \neq a.\mathrm{b} + \mathrm{b}.a$$

In fact, after the delay imposed by *F*, the residual lifetime of *G* has to be computed in order to correctly determine the remaining delay before *b* occurs (refer to Figure 1). To overcome this, the idea is to make a distinction between three activities: (i) starting a delay, (ii) finishing a delay, and (iii) the occurrence of immediate actions. This separation has been brought up by D'Argenio, Katoen and Brinksma [29],[33], they denote their process algebra by ♠. A similar distinction has been made in GSMPA [17].

For more information and for a complete characterization of the algebra, refer to [29],[33],[61].

Once arbitrary probability distributions are allowed, the underlying stochastic process does not have to satisfy the Markov property. It can be shown [17],[61] that the underlying stochastic process is in fact a generalized semi-Markov process (GSMP) (for the case of deterministic process algebras).

2.6.3. Process Algebras: Pros and Cons

- The time required for model construction is often greatly reduced over Markov-like processes. The component-based approach greatly simplifies the task of model construction.
- Basic performance measures could be derived from the solution of the underlying stochastic process without detailed knowledge of the algebra. Generation of the underlying process is formally defined based on the operational semantics of the language. For the case of Markov processes, tools exist to do this automatically [53],[54], and deriving performance is handled the same way as in the Markov process case. However, the identification of states of interest could be easier using the process algebra.
- Stochastic process algebra models bring several attractive features to performance modeling, among these is the compositional structure of these models. It not only aids

model construction (by focusing on one component of the model at a time rather than the whole model) but could also be exploited during model solution [54],[61].

- State space explosion and problem size are the major problem in the calculation of steady state distribution of the underlying process. Extensive research has been dedicated to finding efficient approaches to dealing with the problem [9].

Chapter 3: Non-Markovian Analysis

3.1. Analysis Approaches for non-Markovian Models

With the aim of providing the modeler with more general models, various models have been discussed in the literature. In [28] Cumani presented a model in which every transition is assigned a PH [70] distributed firing time. The steady state and transient state solution for such models was also presented. In [3] the authors presented a model in which exponential and deterministic firing times are allowed with the restriction that at most one deterministic transition can be active at any time. Only the steady state solution was presented in [3]. The transient state probability of these models by the method of embedded regenerative process was then presented in [21]. The method of embedded regenerative process was then extended in [20] to cover any general distribution (not only deterministic) with the restriction that at most one non-exponential distribution could be active at any time (the enabling restriction). In [44], German et al. derived the steady state probabilities of the same model using another method, the method of supplementary variables [26]. The method was then generalized to the transient state analysis in [45].

In what follows we give a brief review for each of these approaches: the method of embedded variables, the method of embedded regenerative process, and the method of continuous phase type distribution PH, more information can be found in [8],[40],[45],[65].

3.2. Method of Supplementary Variables

The method of supplementary variables is well known in queuing literature [26]. It was originally proposed in the context of non-Markovian processes by German et al. in [42],[44], then more general execution policies were discussed in [43]. The method of supplementary variables has been applied to GSMPs $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ with the property that, in each state, at most one enabled event (active event) can have a non-exponential distribution, while all other enabled events are exponentially distributed, a property known as the enabling restriction.

Let $a(t)$ be the age of the only enabled non-exponential event at time t, if any. The enabling restriction implies that if $X(t)$ is the state of the GSMP G at time t, then the new process formed from $X(t)$ and the supplementary variable $a(t)$, i.e. $(X(t), a(t))$, satisfies the memoryless property. The new process has an uncountable state space $S \times \Re$ [44] ($\Re$ is the set of non-negative real numbers), it can be analyzed using the method of supplementary variables as discussed in [44]. The solution approach is hereby briefly summarized following the concepts in [45]:

If $\{g_1, ..., g_n\}$ is the set of non-exponentially distributed events, then the state space S can be partitioned into $n+1$ disjoint sets: $\{S_i, i \in \{1, ..., n+1\}\}$ where $S_i, i \in \{1, ..., n\}$ is the set of states in which event g_i is active, and S_{n+1} is the set of states in which no general transition is enabled. Note that such a partition exists because at most one g_i can be active at any point in time. The probability of being in state s at time t is $\Pi_s(t) = \Pr ob\{X(t) = s\}$. If $s \in S_i, i \in \{1, ..., n\}$, and if the distribution associated with event g_i is $F_i(x)$ then we define the *instantaneous age rate* $h_s(t,x)$ as the conditional firing rate of g_i at time *x* when we are in s at time *t*, given that g_i does not fire before time *x:*

$$h_s(t,x) = \frac{\Pr ob(X(t)=s, x<a(t)\le x+dx)}{dx} \; \frac{1}{1-F_i(x)}$$

where $\Pr ob(X(t)=s, x<a(t)\le x+dx)$ is the probability that g_i will expire when its age is between x and $x+dx$, and $\frac{1}{1-F_i(x)}$ is the probability that g_i does not fire in the interval $[0,x]$. And we define $h_s^{g_i}(t,x)=h_s(t,x)$ if $s\in S_i$ and 0 otherwise, and the vector $h^{g_i}(t,x) =< h_s^{g_i}(t,x), s\in S_i >$.

The state transitions of the stochastic process are given by the following matrices:

- Matrix Q^{g_i} is defined over the set $S_i \times S_i$ where the entry ${Q^{g_i}}^{s,s'}, s\ne s'$, is the rate of transition from state s (in which the general transition g is active) to state s' provided that the transition is exponential. While ${Q^{g_i}}^{s,s}$ is the negative sum of all rates of exponential state transitions out of state s.

- Matrix Δ whose entry $\Delta_{s,s'}$ is the probability of moving from state s to state s' upon the firing of a general transition g_i, i.e. $\Delta_{s,s'} = \Pr ob\{$ the next state is $s' |$ the current marking is s and transition g_i fires$\}$.

With the above definitions, the age rate vector can be described by the following differential equation:

$$\frac{\partial}{\partial t} h^g(t,x) + \frac{\partial}{\partial x} h^g(t,x) = h^g(t,x) Q^g \qquad (1)$$

The transient state probability vector $\Pi(t)$ can be calculated in partitioned form from the age rate vector using the following facts:

The different ways to reach a state in S_{n+1} are:

(i) By the firing of an exponential transition which results in a state in S_{n+1}.

(ii) By the firing of a general transition which results in a state in S_{n+1}.

(iii) By the disabling of a general transition which results in a state in S_{n+1}.

And the different ways to reach a state in S_i are:

(i) By the firing of an exponential transition which results in a state in S_i.

(ii) By the firing of a general transition g_j which results in a state in S_i.

(iii) By the firing of an exponential transition which disables the active general transition g_j and results in a state in S_i (i.e. event g_i is enabled).

The complete system of equations is presented in [40]. A numerical analysis of the equation system is possible by discretization. The method of supplementary variable has a worst case of $O(c\sum_{i=1}^{n} tq^{g_i} |S_i|^2)$ time complexity and $O(|S_{n+1}| + c\sum_{i=1}^{n} |S_i| + \sum_{i=1}^{n} |S_i|^2)$ space complexity where c denotes the time for integral calculation, and q^{g_i} is the absolute maximum diagonal entry for Q^{g_i}. For more information, refer to [40].

3.3. Method of Embedded Regenerative Processes

This technique applies to Markov regenerative processes, MRGP. An MRGP is a GSMP with infinitely many regeneration points, i.e. points where the process satisfies the memoryless property. Because of this property, the analysis of a MRGP can be split into independent sub-problems given by sub-processes starting and ending at a regeneration point. Following this method, transient analysis for systems satisfying the enabling restriction is described in [11],[20],[65].

Definition 3.1: Markov renewal sequence [62]

A Markov renewal sequence is defined as the sequence of pairs of random variables (X_n, θ_n) (usually X_i represents the state of the process that was entered at time θ_i) for which the following properties hold:

$$P\{X_{n+1} = j, \theta_{n+1} - \theta_n \leq x \mid X_n = i, \theta_n, X_{n-1}, \theta_{n-1}, ..., X_0, \theta_0\} =$$

$$P\{X_1 = j, \theta_1 \leq x \mid X_0 = i\}$$

According to the above definition, the current state of the process alone determines probabilistically the next state and the duration of time in the current state.

Given a stochastic process $Y(x)$, if a Markov renewal sequence (X_n, θ_n) is embedded in $Y(x)$, i.e. if the behavior of $Y(x)$ between instants θ_n and θ_{n+1} is of any kind, but at θ_{n+1} $X_{n+1} = Y(\theta_{n+1})$, then $Y(x)$ is called a Markov regenerative process.

Definition 3.2: MRGP [62]

A stochastic process $Y(x)$ is said to be a Markov regenerative process, MRGP, if there exists an embedded Markov renewal sequence (X_n, θ_n) such that

$$P\{Y(\theta_n + x) = j \mid Y(u), 0 \leq u \leq \theta_n, X_n = i\} = P\{Y(\theta_n + x) = j \mid X_n = i\}$$

$$= P\{Y(x) = j \mid X_0 = i\}$$

So MRGPs behave like a Markov process relative to instants θ_n, these instants are known as regeneration instants. But between these instants, the process can evolve in any way. From an intuitive point of view, it can be said that there are instants $\theta_0, \theta_1, ..., \theta_n, ...$ between which the behavior of the process is not affected by its previous history. Moreover, each of the cycles can be studied as if the point of regeneration from which the process is examined where $\theta_0 = 0$.

As described in [62] two quantities are capable of describing the evolution of the MRGP are defined:

- The local kernel $E(t): E_{ss'}(t) = P(Y(t) = s' \wedge \theta_1 > t \mid X_0 = s)$ describes the evolution of the process between two regeneration instants, and
- the global kernel $K(t): K_{ss'}(t) = P(X(t) = s' \wedge \theta_1 \leq t \mid X_0 = s)$ describes the evolution of the process at the regeneration instants themselves.

So if $V(t) = [V_{ss'}(t)] = [\Pr ob\{Y(t) = s' \mid Y(0) = s\}]$ denotes the transition probability of the MRGP, then the transient state probabilities can be obtained by solving the following equation [45]:

$$V_{ss'}(t) = E_{ss'}(t) + \sum_r \int_0^t dK_{sr}(x) V_{rs'}(t-x) \qquad (1)$$

Or its Laplace transform domain [40]

$$V^{\sim}(x) = [I - K^{\sim}(x)]^{-1} E^{\sim}(x) \qquad (2)$$

A solution can be obtained by numerically integrating equation (1), or by a combination of numeric and symbolic computation for equation (2). In both cases, the complexity of the solution of the above equation limits the applicability of this technique to MRGPs implementing the enabling restriction [40],[45]. With MRGPs implementing the enabling restriction, the regenerative method has $O(|S|^2)$ space complexity and $O(|S|^4)$ time complexity in the worst case [40].

3.4. Continuous Phase Type Distribution

This technique consists of approximating general distributions by a series/parallel combination of exponential distributions, thus transforming the process into a Markovian one.

An interesting class of distributions that serves this purpose are the PH distributions [70]. PH distributions can be defined as "the distributions of absorption time in a CTMC with a single absorption state" [8]. PH distributions can approximate any distribution arbitrarily close. Many tools are available to find the PH distribution that approximates any given general distribution. After approximating all distributions of events in the process using PH distributions, the process is expanded into a CTMC. The expansion algorithm can be performed automatically by a computer program. Exact results are obtained only when the firing times of the original process are PH-distributed. For recent updates on this method refer to [8].

3.5. Contributions Outline

The first two methods presented in this chapter deal with the case where events with non-exponentially distributed durations are mutually exclusive. Imposing this restriction leads to algorithms with reasonable costs while going beyond the restriction is one of the most challenging open issues in the field [10],[38],[41],[55]. However for events with deterministic durations, efficient numerical analysis can be found [42]. For a comparison of these approaches refer to [40],[45]. The continuous phase type distribution technique is not restricted to Markov regenerative processes, however, exact results are only obtained when all firing times of the original process are PH-distributed.

In this thesis, we will extend the class of solvable GSMPs by allowing several generally distributed events to be enabled at any time. However, we impose the restriction that every cycle $C = s_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{e_{n-1}} s_n \xrightarrow{e_n} s_1$ in the GSMP must either be near semi-Markovian or regenerative. These properties are defined as follows:

1. cycle C is **near semi-Markovian (NSM)**, if every state s_i in the cycle has the property that given any event g that starts its lifetime in a state s_j of C, if g is active in s_i then g restarts its lifetime in s_i. Intuitively, when taken as a separate entity from the GSMP, C becomes a semi-Markovian process. An NSM cycle is called **near Markovian** (NM), if all the events activated inside the cycle (i.e. the set of events $K(C)=\bigcup_{i=1}^{n} K(s_i)$) have exponentially distributed delays. Formally, C is NSM iff for all $s_i \in C$, if $(g \in A(s_i) \wedge (\exists j \neq i; g \in A(s_j))$ then $g \in K(s_i)$

2. Cycle C is **regenerative** (REG), if there exists a regenerative state s_i in the cycle (i.e. the GSMP satisfies the Markovian property at the time state s_i is entered). Formally, there exists $s_i \in C$, such that $A(s_i) = K(s_i)$.

GSMPs whose cycles are either NSM or REG are referred to as near-regenerative generalized semi-Markov processes, NRGSMP. NRGSMPs are more general than the GSMP's implementing the enabling restriction which we will abbreviate as EGSMPs. In fact, among other restrictions, the only cycles allowed in a EGSMP are either regenerative or near-Markovian. However, an important class of GSMP's is not covered by NRGSMPs; an example is the queue G/G/1 of size 3 shown in Figure 4. Note that the cycle $2 \xrightarrow{s} 1' \xrightarrow{a} 2$ is neither regenerative nor near semi-Markovian. In fact, all GSMPs with at least one cycle satisfying both points below:

- The cycle has no regenerative states, and
- There exists a transition $s \xrightarrow{e} s'$ such that $e \notin K(s)$

are not NRGSMPs, and the cycle $2 \xrightarrow{s} 1' \xrightarrow{a} 2$ of Figure 4 satisfies both of these points. NRGSMPs will be formally defined in Chapter 4 .

In finding the steady state probabilities for NRGSMPs, we will present an algorithm to transform the NRGSMP into a semi-Markov process (SMP) while preserving *steady-state simulation*, which enables us to determine the steady state probability of the NRGSMP from that of the SMP constructed. The algorithm will be presented in full details in Chapter 4. The chapter will also include a comparison between the different methods.

The method described above could generate semi-Markov processes with large state spaces. For that reason, Chapter 5 introduces a method to remove states from GSMPs while preserving the distribution of time needed to travel between non-deleted states and preserves the transient state probabilities for a subset of the states of the automata as well. Chapter 6 deals with the issue of state space explosion of the SMP created; it deals with the problem by introducing a new simplification technique for semi-Markov processes. The technique deletes states from the SMP while preserving the average time to travel between non-deleted states, or what we call mean passage-time equivalence. Chapter 7 presents an application for Chapters 4,5 and 6, a case study, and presents the procedure needed to determine whether a GSMP is an NRGSMP. Finally, Chapter 8 provides our conclusions and suggestions for future research directions.

Chapter 4: A Deeper Look into Non-Markovian Analysis

4.1. Introduction

In this chapter, we will present a new class of processes: Near-Regenerative GSMP's or NRGSMP. The NRGSMP's extend the class of solvable GSMPs by allowing several events with a generally distributed lifetime to be enabled at the same time. However, they satisfy the restriction that every cycle $C = s_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{e_{n-1}} s_n \xrightarrow{e_n} s_1$ must either be **near semi-Markovian** (NSM) or **regenerative** (REG), recall from Section 3.5 that

1. cycle C is **near semi-Markovian (NSM)**, if every state s_i in the cycle has the property that given any event g that starts its lifetime in a state s_j of C, if g is active in s_i then g restarts its lifetime in s_i. Intuitively, when taken as a separate entity from the GSMP, C becomes a semi-Markovian process. An NSM cycle is called **near Markovian** (NM), if all the events activated

inside the cycle (i.e. the set of events $K(C) = \bigcup_{i=1}^{n} K(s_i)$) have exponentially distributed delays. Formally, C is NSM iff for all $s_i \in C$, if $(g \in A(s_i) \wedge (\exists j \neq i; g \in A(s_j))$ then $g \in K(s_i)$

2. Cycle C is **regenerative** (REG), if there exists a regenerative state s_i in the cycle (i.e. the GSMP satisfies the Markovian property at the time state s_i is entered). Formally, there exists $s_i \in C$, such that $A(s_i) = K(s_i)$.

These properties will be formally defined in the next section.

For finding the steady state probabilities for NRGSMPs, we will present an algorithm to transform the NRGSMP into a semi-Markov process (SMP) while preserving *steady-state simulation*, a simulation that enables us to determine the steady state probabilities of the NRGSMP from that of the SMP constructed. To transform the NRGSMP G into such an SMP G' we need to calculate, for every state s, the distribution of the remaining lifetime of every active event e of s given that G has been running for a sufficiently long time. We call this distribution the "average residual lifetime distribution" and we denote it by $Av\mathrm{Re}s(e,s)$. We call the distribution "average residual lifetime distribution" because it is the expected distribution of event e in state s regardless of the trace followed to reach state s. To explain the transformation and the equivalence informally, let s be a non-regenerative state in G, and let Γ be the set of all traces in G from the starting state s_0 to s. Note that Γ could be infinite. We construct G' as follows:

- We divide Γ into distinct subsets $\{\Gamma_1, \Gamma_2, \ldots \Gamma_n\}$ such that each Γ_i has the following property:
    - if we reach state s by following a trace from Γ_i, the average residual lifetime for the active events of s can be calculated (Note that given two traces T and T' in Γ_i, $Av\mathrm{Re}s(e,s,T)$ may not be equal to $Av\mathrm{Re}s(e,s,T')$. However, given that a trace from Γ_i was followed to reach state s, then the average

residual lifetime for the active events in state s, which is the expected distribution associated with the active events in s can be calculated). We will see in Section 4.3 how such Γ_i's look like.

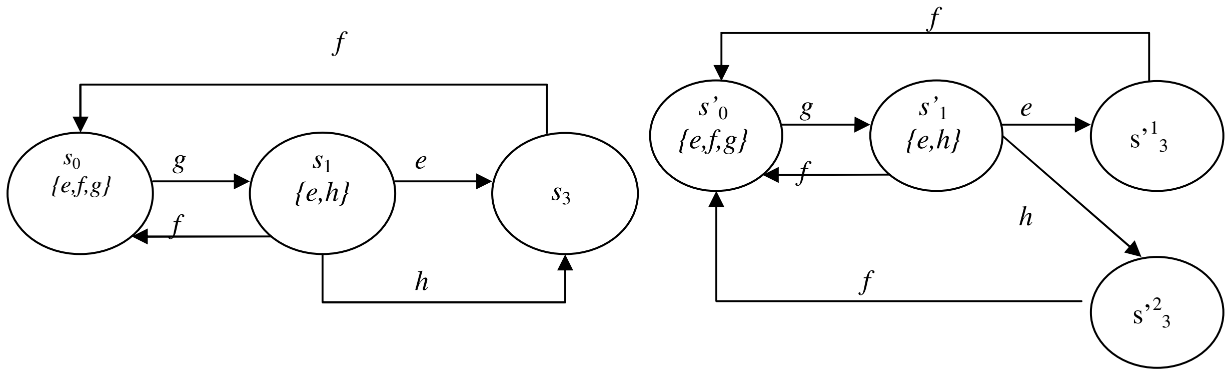
- Then we create n copies of state s: $\{s_1,...,s_n\}$ one for each subset Γ_i. We say that state s is split into n states. And we modify the GSMP G such that every set of traces Γ_i leads to state s_i, i.e. the set of traces from the starting state s_0 to s_i becomes Γ_i. (Note that this step might involve splitting states other than s). The active event e in s_i is then assigned a distribution equal to $Av\mathrm{Re}s(e,s_i)$. We identify the modified process as G'.

Based on the above construction, it will be shown that G' steady-state simulates (or s-simulates) G. One of the properties of the simulation is that, given G and G' have been running for a sufficiently long time, the distribution of the sojourn time in state s in G, denoted by $\zeta^G(s)$, is equal to $\sum_{i=1}^{n}\frac{P^{G'}(s_i)\zeta^{G'}(s_i)}{\sum_{j=1}^{n}P^{G'}(s_j)}$, where $P^{G'}(s_i)$ is the probability of being in state s_i in G' and $\zeta^{G'}(s_i)$ is the sojourn time in state s_i in G'.

Another property of the s-simulation is that if a GSMP s-simulates another, then the two GSMPs are bisimulation equivalent from a functional point of view, i.e. if we neglect the time distributions associated with the events and consider only the order of events labels, the two GSMPs become bisimulation equivalent.

To illustrate this, we consider the two GSMPs G and G' depicted in Figure 5. We consider the following relation R: $s_0Rs'_0$, $s_1Rs'_1$, $s_3Rs'^1_3$, and $s_3Rs'^2_3$ (i.e. state s_3 was split into two states s'^1_3 and s'^2_3) . Note that the two GSMP's are bisimulation equivalent from a functional point of view.

While the algorithm presented in this chapter provides a theoretical solution for all NRGSMP's, its applicability is restricted because of its exponential time and space complexities. However, we have identified a subset of NRGSMPs for which the algorithm has $O(n^2)$ space complexity and a polynomial time complexity. This subset contains all the EGSMP's, and exceeds them to include a subset of the NRGSMPs that are not EGSMPs (refer to Figure 10 in Section 4.3.3).



(a) GSMP G **(b)GSMP G'**

Figure 5. Two GSMPs

The algorithm for transforming an NRGSMP into an SMP works by transforming all non-regenerative states into regenerative states by splitting states as required. The algorithm will be presented in Section 4.4 after defining NRGSMPs and explaining the intuition behind the transformation in Section 4.2. Some needed definitions and preliminary results will be presented in Section 4.3. And finally, the complexity of the algorithm is discussed in Section 4.5.

4.2. Preliminary Definitions and Algorithm Overview

In this section, we present the formal definition for NRGSMPs followed by the algorithm overview. We start first with some preliminary definitions.

4.2.1. Preliminary definitions

Definition 4.1: Regenerative state

A *regenerative state s* in a GSMP is a state with the property: $A(s) = K(s)$.

Definition 4.2: Trace, regenerative trace, single regenerative trace, simple trace, path, execution sequence and cycle

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP. Assume that the set of transitions $\mapsto$ contains n transitions denoted by $t_1,...,t_n$

- A *trace* of G is a finite sequence of transitions $t_{i_1}, t_{i_2},...$ such that transition t_{i_j} leads to a state from which transition $t_{i_{j+1}}$ is possible (has a non-zero probability).
- A *regenerative trace* of G is a trace $t_{i_1}, t_{i_2},...$ such that t_{i_1} is a transition out of a regenerative state.
- A *single regenerative trace* of G is a regenerative trace that contains exactly one regenerative state.
- A *simple trace***,** also referred to as *path*, is a trace with no cycles (i.e. no state appears twice in the trace)
- An *execution sequence Ex* is a tuple of the form $Ex = (T, < x_1,...,x_n >)$ where $T = s_1 \xrightarrow{e_1} s_2 ... \xrightarrow{e_{n-1}} s_n$ is a regenerative trace and x_i is the time spent in state s_i, or what is known as sojourn time in state s_i denoted by $\varsigma(s_i)$ for all $i \in \{1,...,n\}$.
- An *n-cycle* in a GSMP is a trace of the form $T = s_1 \xrightarrow{e_1} s_2 ... \xrightarrow{e_{n-1}} s_n \xrightarrow{e_n} s_1$ where $s_i \neq s_j$ for $i \neq j$, $i, j \in \{1,...,n\}$.

Definition 4.3: Near-Markovian, near Semi-Markovian, and regenerative cycles

An *n*-cycle $c = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{n-1}} s_n \xrightarrow{e_n} s_1$ in a GSMP is said to be:

- *Near semi-Markovian (or NSM),* if

  (a) All events on the cycle, i.e. $\{e_1,\ldots,e_n\}$, are initialized just before they are executed, formally, $e_j \in K(s_j)$ for all $j \in \{1,\ldots,n\}$ and

  (b) If a non-cycle event g (i.e. $g \notin \{e_1,\ldots,e_n\}$) is initialized in some state in the cycle, then this event can only be active in a state on the cycle provided it is initialized there. Formally if there exists a state s_i, $i \in \{1,\ldots,n\}$, such that $g \in K(s_i)$ and $g \notin \{e_1,\ldots,e_n\}$ then $\left(\forall j \in \{1,\ldots,n\}, g \in A(s_j) \Rightarrow g \in K(s_j)\right)$

- *Regenerative (REG),* if at least one of its states is a regenerative state.

- An NSM cycle is called *near-Markovian (NM)* if all events in $K(s_i), i \in \{1,\ldots,n\}$ are exponentially distributed.

An example of a near semi-Markovian cycle is shown in Figure 6. If e, e', e'', s were exponentially distributed (in all states they are initialized in) then the cycle becomes near-Markovian.

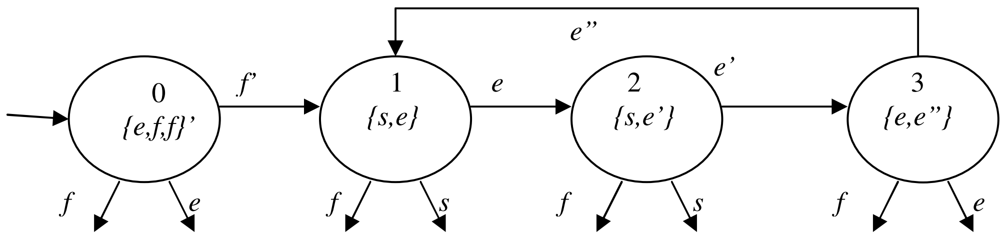


Figure 6. Part of a GSMP

Definition 4.4: NRGSMP

A *near-regenerative generalized semi-Markov process*, NRGSMP, is a GSMP with the following restriction: All the cycles in the process are NSM or REG cycles.

Note that, although these restrictions are strong, previous work [42],[44],[11],[20],[65] imposed the so-called "enabling restriction" which implies that only regenerative and NM cycles are allowed, so NSM cycles were not allowed, this in addition to the fact that only one non-exponentially distributed event could be active at any given time. Figure 3 and Figure 10 show examples of NRGSMPs. The NRGSMP in Figure 10 has only regenerative cycles, yet it does not satisfy the "enabling restriction", in other words it is not an EGSMP. The next Theorem describes properties of traces in an NRGSMP.

4.2.2. Algorithm overview

To transform a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ into an SMP we need to transform every non-regenerative state s in G into a regenerative state. To be able to transform s into regenerative, we will assume that we know the time l elapsed since the GSMP started running until state s was entered.

Let s be a non-regenerative state in G; for the sake of simplicity, assume that $A(s) = \{e\}$ and $K(s) = \phi$. To transform s into a regenerative state, we need to find the residual lifetime for events that are active in s, or in other words, the probability that event e will occur within x time units from entering s given that the GSMP has been running for l time units when state s was entered, denoted by $Av\mathrm{Re}\,s^l(e, s)(x)$.

Let us assume that we know the single regenerative trace that was followed to reach s: $T = s_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{m-1}} s_m = s$ and the times x_i spent in each of the states s_i for all

$i \in \{1,...,m\}$: $(T, < x_1,...,x_m >)$. then we can easily calculate the "conditional probability that e will occur in x time units after entering s". We call this probability distribution *the residual time of event e in state s given* $(T, < x_1,...,x_m >)$ given that the GSMP has been running for l time units when state s was entered, and denote it by $\mathrm{Re}\, s^l(e, s, (T, < x_1,...,x_m >))$. Note that $l \geq x_1 + ... + x_m$ (and $l = x_1 + ... + x_m$ if $s_1 = s_0$).

If the single regenerative trace T that was followed to reach s is known but not the time spent in the different states in the trace, then the "probability that e will occur in x time units after entering s given T was followed and given that the GSMP has been running for l time units when state s was entered" is denoted by $\mathrm{Re}\, s^l(e,s,T)(x)$. Assume for simplicity that $T = s' \xrightarrow{e'} s$, then $\mathrm{Re}\, s^l(e,s,T)(x)$ can be calculated from the distribution of the residual time of the events in state s' as follows:

$$\mathrm{Re}\, s^l(e,s,T)(x) = \int_0^l \frac{P(\varsigma(s') = x' \mid T)}{dx'} P(e_{s'}[x', x'+x] \mid \varsigma(s') = x' \wedge T)$$

Where $e_{s'}[x', x'+x]$ is the fact that event e occurs in the interval $[x', x'+x]$ since entering s', and T stands for the fact that transition T occurred to reach state s. We will prove in Theorem 4.6 that the above formulae is a function of the distribution of the residual time of the events in state s'.

Consider for example State 1 in Figure 3, note that event a is active in this state,. Note also that this state can be reached through infinitely many single regenerative traces $\Gamma_1 = \{T_1^n = 0 \xrightarrow{d} 1(\xrightarrow{r} 2 \xrightarrow{s} 1)^n \mid n \in \mathrm{N}\}$ where n indicates the number of times cycle $1 \xrightarrow{r} 2 \xrightarrow{s} 1$ is executed, Note that every trace T_1^n in Γ_1 will give a different value for $\mathrm{Re}\, s^l(a,1,T_1^n)$.

Now, assume that the single regenerative trace that was followed to reach s belongs to a set of single regenerative traces Γ_s, then the "probability that e will occur in x time units after entering s given that a trace in Γ_s was followed and given that the GSMP has been running for l time units when state s was entered" is denoted by $Av\operatorname{Re}s^l(e,s,\Gamma_s)(x)$. Note that $Av\operatorname{Re}s^l(e,s,\Gamma_s)$ is the average over $\{\operatorname{Re}s^l(e,s,T), T\in\Gamma_s\}$ taking into consideration the different probabilities of the regenerative traces in Γ_s, in other words, $Av\operatorname{Re}s^l(e,s,\Gamma_s)=\sum_{T\in\Gamma_s}P(T)\operatorname{Re}s^l(e,s,T)$ where $P(T)$ is the probability that the regenerative trace T occurs given that a trace from the set Γ_s will occur.

If Λ_s is the set of all single regenerative traces leading to s, let ∇_s be the set of all regenerative states that may lead to state s, and for every $r\in\nabla_s$, let $T_r\subseteq\Lambda_s$ be the set of all traces that start with state r and lead to state s ($T_r\subseteq\Lambda_s$), then

$$Av\operatorname{Re}s^l(e,s,\Lambda_s)(x)$$

$$=\sum_{r\in\nabla_s}\frac{P_r}{\sum_{r\in\nabla_s}P_r}Av\operatorname{Re}s^l(e,s,T_r)(x)$$

Where P_r is the probability of being in state r given that the GSMP has been running for a sufficiently long time, and $\frac{P_r}{\sum_{r\in\nabla_s}P_r}$ is the probability P_r normalized over the set of regenerative states ∇_s, and $Av\operatorname{Re}s^l(e,s,T_r)$ is as defined earlier.

Note: In Section 4.3.4, we will calculate $Av\operatorname{Re}s^l(e,s,T_r)$ as a function of l and x. However, throughout this thesis, and in transforming the GSMP into an SMP, we will assume that the GSMP is in steady state, in other words, we will only consider the case where $l=\infty$ as we will be extracting performance measures assuming steady state. In that case, $Av\operatorname{Re}s^l(e,s,T_r)$ will be denoted by $Av\operatorname{Re}s(e,s,T_r)$.

The questions that come to mind now are how and under what restrictions can we calculate $AvRes(e,s,\Lambda_s)$? As discussed earlier, $AvRes(e,s,\Lambda_s) = \frac{\sum_{r\in\nabla_s} P_r AvRes(e,s,T_r)(x)}{\sum_{r\in\nabla_s} P_r}$, but P_r is a function of the SSP of state r, which is not always possible to get.

We will prove in Section 4.3.4 that we can calculate $AvRes(e,s,\Lambda_s)$ in the following three cases:

1. If $\Lambda_s = \{T\}$, in other words if $AvRes(e,s,\Lambda_s) = Res(e,s,T)$.

2. If all traces in Λ_s start with the same regenerative state r (i.e. all the traces travel from r to s), then let N be the sub-process of G with state space S_N formed from the states belonging to Λ_s-{ r }, and its transitions are the transitions between the states in S_N. Then if for all r', r'' in S_N the following is satisfied

   i. if $t : r' \xrightarrow{f} r''$ belongs to N, then $f \in K(r')$ and

   ii. if $e \in K(r'') \cap A(r')$ then $e \in K(r')$ and,

   iii. $A(r') - K(r') \subset A(r)$

   then N is called an embedded semi-Markov process; ESMP, intuitively, the points above mean that the sub process N is an SMP. In such case, all events, say f, that are initialized in r and are active in s, $AvRes(f,s,\Lambda_s)$ will be calculated using the quantity $\lim_{x\to\infty} P(s(x) \mid r(0) \wedge \Lambda_s)$ where $P(s(x) \mid r(0) \wedge \Lambda_s)$ is the probability of being in state s at time x given we entered state r at time 0 and given that Λ_s was followed from r to s. The value of $P(s(x) \mid r(0) \wedge \Lambda_s)$ will be calculated using the fact that N is an SMP. This will be presented in details in Section 4.4.2.

3. A combination of the above, i.e. Λ_s is of the form: $\Lambda_s = T_{1s_1}^{s_2} N_1 *_{s_2}^{s_3} T_{2s_3}^{s_4} N_2 *_{s_4}^{s_5} \ldots T_{n s_{m-2}}^{s_{m-1}} N_n *_{s_{m-1}}^{s_m}$ where:

    a. s_1 is regenerative

    b. $T_{is_j}^{s_k}$ is a path, from s_j to s_k and

    c. N_i is an ESMP

    d. $N_i *_{s_j}^{s_k}$ is the set of traces from s_j to s_k belonging to N_i

Such a set Λ_s will be called a *single-AvRes* set and will be defined in Section 4.3.

The average residual distributions for the events in state s where Λ_s is of the form specified under point (3) are calculated as follows:

- We first calculate the average residual distribution for all events in $T_{1s_1}^{s_2}$ sequentially (starting from s_1), thus making all the states in $T_{1s_1}^{s_2}$ regenerative
- The next step would be to calculate the average residual distributions for all active events in the states of N_1 using the average residual distributions of the events in s_2 that were calculated in the previous step
- We continue with $T_{2s_3}^{s_4}$, and so on ….

Consider again $\Gamma_1 = \{T_1^n = 0 \xrightarrow{d} 1(\xrightarrow{r} 2 \xrightarrow{s} 1)^n \mid n \in \mathrm{N}\}$ of Figure 3. We will see in Section 4.4 that $1 \xrightarrow{r} 2 \xrightarrow{s} 1$ satisfies condition (2) above, and as a result we will calculate $Av\mathrm{Re}s(a,1,\Gamma_1)$.

Now, let Γ_s be the set of all single regenerative traces leading to s (recall that these traces have exactly one regenerative state). The conditions set on the cycles of the GSMP will allow us to partition the set of regenerative traces $\Gamma_s : \{ \Gamma_i, i \in I \}$, such that each Γ_i is a *single-AvRes set*, i.e. a set where we will be able to calculate $Av\operatorname{Re}s(e,s,\Gamma_i)$ for each $i \in I$. Now given $\Gamma_i, i \in I$, we transform G into a GSMP G_s by splitting (or unfolding) state s into several states $s_i, i \in I$ such that each s_i is only reached through the traces with postfixes in Γ_i. The algorithm in all its details will be presented in Section 4.4. Then, after splitting every non-regenerative state as described, the resulting process G' and G are shown to be structurally bisimilar (refer to Definition 9 in Section 4.3). The process G' will be called *Hidden Markov Regenerative Process* (HMRP) and will be presented in Section 4.3.

The last step is to calculate $Av\operatorname{Re}s(e,s_i,\Gamma_i)$ for each $i \in I$, and assign it to $F_{s_i}(e)$. The resulting GSMP would s-simulate G. This simulation implies that the probability of leaving state s within a certain time after reaching it, given that we reached s through a single regenerative trace in Γ_i, is preserved.

So given the NRGSMP G, assume that $L = \{s^1,...,s^l\}$ is the set of states that need to be transformed into regenerative states. As described above, we need two steps to transform it into an SMP:

1. We will first transform G into what we call a Hidden Markov regenerative process (HMRP) G' while preserving structural bisimulation.

2. The next step is to calculate $Av\operatorname{Re}s(e,s^i)$. The states s^i can then become regenerative by assigning to events e in s^i the distribution $Av\operatorname{Re}s(e,s^i)$. After transforming all non-regenerative states of G' into regenerative ones, we obtain an SMP G'' that s-simulates the original one. We will see in Section 4.3 how to get steady state probabilities of G from those of G''.

4.3. Definitions, Illustrations and Preliminary Results

In this section, we review a concept from stochastic systems: structural bisimulation. And we introduce some new concepts: residual time, single-*AvRes* set of traces, hidden Markov regenerative process, steady-state simulation, embedded SMP, and in-borders of an embedded SMP.

4.3.1. General Definitions and Results

Definition 4.5: Complete sub-GSMP, in-border, embedded SMP

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP,

- A *complete sub-GSMP* of G is a GSMP with no starting state: $M = (S', \mathrm{E}, F, A, \mapsto', K)$ where $S' \subset S$ and $\mapsto' = \mapsto \cap (S' \times E \times S')$, in other words, $\mapsto'$ is the biggest subset of $\mapsto$ linking all states in S'. A state r is said to be an *in-border* of M if $r \in S - S'$, and if there exists a transition: $r \xrightarrow{e} s$ for some $s \in S'$ (note that components $E, F, A,$ and K are not equal to the components in the original GSMP G, in fact these are their restriction to the state set S', but we use the same notation for simplicity).

- An *embedded semi-Markov process (ESMP)* M of G, $M = (S', \mathrm{E}, F, A, \mapsto', K)$, is a complete sub-GSMP of G such that:

  1. If $K(S') = \bigcup_{s' \in S'} K(s')$ is the set of all events initialized in the states belonging to the ESMP, then for all $s' \in S'$, and for all $e \in K(S')$, either $e \in K(s')$ or $e \notin A(s')$

  2. $A(s') - K(s') = A(s'') - K(s'')$ for all $s', s'' \in S'$, and

3. S' is the maximal set of states with the above properties.

- Let $M = (S', \mathrm{E}, F, A, \mapsto', K)$ be an ESMP, and let $s'_0 \in S'$, then the *ESMP of* s'_0 is the ESMP M having s'_0 as the starting state. Note that the state space of M' becomes the set of states that are reachable from s'_0, in other words, $M' = (S'', s'_0, \mathrm{E}, F, A, \mapsto'', K)$ where S'' is the set of states that belong to S' and that are reachable from s'_0, and $\mapsto''$ is a set of all possible transitions from $\mapsto'$ linking states in S''.

We note the following:

- An ESMP $M = (S', \mathrm{E}, F, A, \mapsto', K)$, when taken as a separate entity is a semi-Markov process: If r is an in-border of M, then an event e that is initialized in r might be active in a state s of M but not initialized in it, i.e. $e \in A(s) - K(s)$, then, from point (2) above, e should never be initialized in any state of the ESMP, and should never occur inside the ESMP, so M satisfies the properties of an SMP.
- An ESMP could be trivial, i.e. consisting of only one state.
- From the above definition, we deduce that if a state $s \in S'$ is regenerative, then all states in S' are regenerative

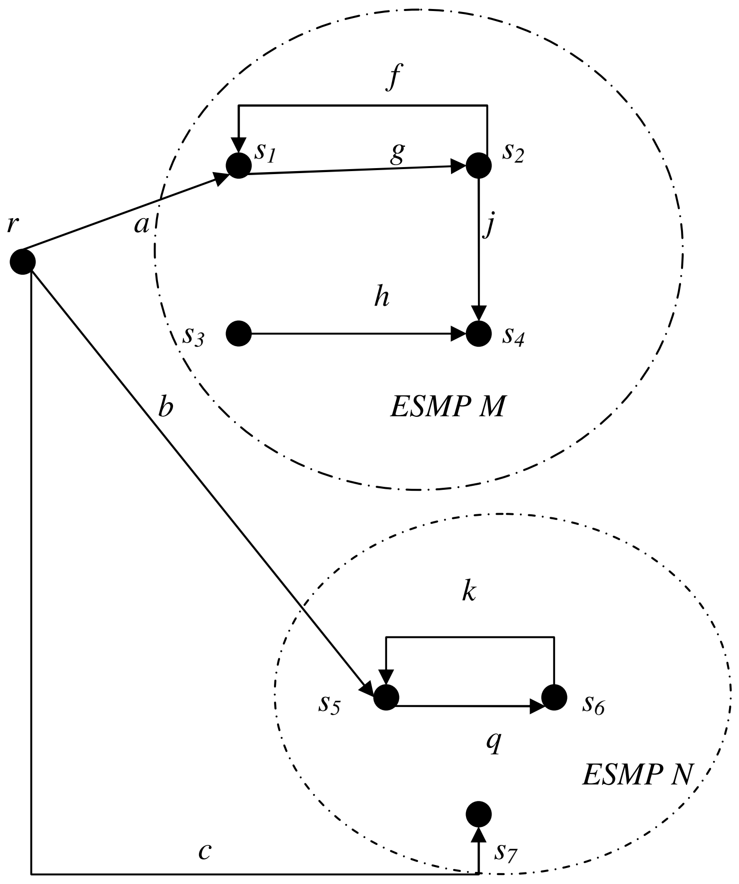


Figure 7. Nest

Figure 12(a) shows an ESMP with its in-border 0-0 corresponding to the GSMP described in Figure 9.

Consider the part of a GSMP shown in Figure 7: We assume that $K(r)=\{a,b,c\}$, $K(s_1)=\{g\}$, $K(s_2)=\{f,j\}$, $K(s_3)=\{h\}$, $K(s_4)=\{\}$, $K(s_5)=\{q\}$ and $K(s_6)=\{k\}$. We also assume that $A(r)-K(r)=\{\}$, $A(s_1)-K(s_1)=A(s_2)-K(s_2)=A(s_3)-K(s_3)=A(s_4)-K(s_4)$ $=\{b,c\}$ and that when events b or c occur in states s_1,s_2,s_3 or s_4 then they lead to a state outside ESMP *M*. And similarly, we assume that $A(s_5)-K(s_5)=A(s_6)-K(s_6)=A(s_7)-K(s_7)=\{a\}$ and when event a occurs in state s_5 or s_6 or s_7 then it leads to a state outside *N*. With the above assumptions in mind, we deduce that we have two ESMPs: ESMP M is composed of four states: $\{s_1, s_2, s_3, s_4\}$, and its in-border is r. And ESMP N is composed of three states s_5, s_6 and s_7 and has the same in-border. The ESMP of state s_1 and the ESMP of state s_2 are both composed of the three states $\{s_1, s_2, s_4\}$, and the ESMP of state s_3 is composed of two states $\{s_3, s_4\}$.

We will see later that given an ESMP and its in-border, then once its in-border is transformed into a regenerative state, the residual distributions for the events in all the states of the ESMP can be calculated using the semi-Markovian properties of the ESMP. This will be explained in detail in Subsection 4.3.4.

Definition 4.6: Nest of a state

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, and let $s \in S$. We define a *nest of* s as the process $N = (S'' \cup \{s\}, s, E, F, A, \mapsto'', K)$, where $M = (S', \mathrm{E}, F, A, \mapsto', K)$ is an ESMP with in-border s, $S'' \subseteq S'$ is the set of all states from S' that are reachable from s, and $\mapsto''$ is the set of all possible transitions from $\mapsto$ linking the states of N: $\mapsto'' = \mapsto \cap (S'' \cup \{s\} \times E \times S'' \cup \{s\})$. If s is not the in-border of any nest, then the nest of s is the sub-process composed of one state s.

Note that the nest of a trivial ESMP is composed of two states only.

Figure 7 depicts two nests of a state r, one is composed of four states $\{r, s_1, s_2, s_4\}$, and the other is composed of states $\{r, s_5, s_6, s_7\}$.

We note that, a nest $N = (S, r, E, F, A, \mapsto, K)$ of a state r is an SMP. In fact, an event $e \in A(r)$ might be active in all states in S but not initialized in these states, i.e. $e \in \bigcap_{s \in S} A(s) - K(s)$, however, e is never initialized in any state of the nest, and does not occur in the nest, so the nest satisfies the properties of an SMP.

Note: if $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ is a GSMP, and $N = (S', \mathrm{E}, F, A, \mapsto', K)$ is an ESMP of G, and r_1 is an in-border for N, then if we let $N(r_1)$ be the process whose traces are of the form tT where $t : r_1 \to s'$ is any trace from r_1 to a state $s' \in S'$, and T is a trace in N starting in state s', then $N(r_1)$ is a nest of r_1.

Lemma 4.1.

1. Every near semi-Markovian cycle is part of an ESMP.
2. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, then the set of all ESMPs in G are distinct. Moreover, every non-regenerative state belongs to exactly one ESMP.
3. Let $M' = (S', \mathrm{E}', F', A', \mapsto', K')$ be an ESMP with in-border s'_0, in a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$. If $t : s'_0 \xrightarrow{e} s$ and $t' : s'_0 \xrightarrow{e'} s'$ belong to M', then either
    - $t = t'$ or
    - e and e' have an exponentially distributed lifetime or
    - e and e' are not active in any state of M'.

Proof.

- Point 1: Straightforward from the definition of a semi-Markovian cycle
- Point 2: Let $M = (S_M, s_0, \mathrm{E}, F, A, \mapsto_M, K)$ and $N = (S_N, s'_0, E', F', A', \to_N, K')$ be two ESMPs in G, assume that a state s belongs to both ESMPs, then we deduce that $A(s) - K(s) = A'(s) - K'(s) = \quad A(s') - K(s') = A'(s'') - K'(s'')$ for all $s' \in S_M$ and $s'' \in S_N$, and. Hence the states in $S_M \cup S_N$ should form an ESMP because of property (3) in Definition 4.5.

- Point 3 is a result of the restriction on ESMPs: $A(s') - K(s') = A(s'') - K(s'')$ for all $s', s'' \in S'$.

∎

Theorem 4.1. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, and for any $s_i \in S$ let $M_{s_i} = (S_i, s_i, \mathrm{E}, F, A, \mapsto_i, K)$ be the ESMP of state s_i. Then we have the following properties:

- Given two states $s_i, s_j \in S$, if $s_j \in S_i$, then $S_j \subseteq S_i$, and if M_{s_i} is strongly connected (i.e. there exists a path between any two states of M_{s_i}) then $S_j = S_i$.
- There exists a subset $S' \subset S$, such that the set $\{M_{s_i} \mid s_i \in S'\}$ has the following properties:

    1. $S_i \cap S_j = \phi$ for all $s_i \neq s_j \in S'$ and
    2. for all $s_i \in S$, $S_i \subseteq S_h$ for some $s_h \in S'$.

    The set $\{M_{s_i} \mid s_i \in S'\}$ will be called *maximal set of connected ESMP parts*.

Proof. Straightforward.

∎

If we divide each of the M_{s_i}'s in the set $\{M_{s_i}, s_i \in S'\}$ in the theorem above into several sub-ESMP's $\{M^j{}_{s_i}, M^j{}_{s_i} \subseteq M_{s_i}\}$ where each $M^j{}_{s_i}$ is strongly connected, then the set $\{M^j_{s_i} : s_i \in S', M^j_{s_i} \subseteq M_{s_i}\}$ will be called *maximal strongly connected set of ESMP parts.*

Definition 4.7: Residual lifetime and average residual lifetime

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP and let $e \in A(s),\ s \in S$. The following definition assumes that the GSMP has been running for l time units when state s was entered.

- The *residual lifetime of e in s after a given execution sequence $Ex = (T, < x_1, ..., x_m >)$ has occurred*, written $\mathrm{Re}\, s^l(e, s, (T, < x_1, ..., x_m >))(x)$ (where $\mathrm{Re}\, s$ is short for Residual), is the probability that event e will occur in x time units after entering s given $(T, < x_1, ..., x_m >)$ was followed.

- The *residual lifetime of e in s after a given regenerative trace T had occurred* (the timings associated with T are not known) , written $\mathrm{Re}\, s^l(e, s, T)$, is the time distribution for event e in state s given that the state was entered through the regenerative trace T.

- Let $\{T_i \mid i \in I\}$, where I is a set of integers, be a set of regenerative traces leading to s, and let $P(T_i)$ be the probability that the regenerative trace T_i occurred given that a trace from the set $\{T_i \mid i \in I\}$ will occur. Then we define:

    1. the average residual time of event e in s given that a regenerative trace from the set $\{T_i \mid i \in I\}$ had occurred, written $Av\mathrm{Re}\, s^l(e, s, \{T_i \mid i \in I\})$, as the time distribution for event e in state s given that a trace in $\{T_i \mid i \in I\}$ just occurred. In other words, $Av\mathrm{Re}\, s^l(e, s, \{T_i \mid i \in I\})(x)$ is the probability that event e will occur within x time units after reaching s given that s was reached by following a trace in $\{T_i \mid i \in I\}$. Note that, $Av\mathrm{Re}\, s^l(e, s, \{T_i \mid i \in I\}) = \sum_{i \in I} P(T_i)\mathrm{Re}\, s^l(e, s, T_i)$. If $\{T_i \mid i \in I\}$ is the set of all possible regenerative traces leading to s, we write $Av\mathrm{Re}\, s^l(e, s)$ instead of $Av\mathrm{Re}\, s^l(e, s, \{T_i \mid i \in I\})$.

    2. $Av\mathrm{Re}\, s^l(e, s, \{T_i \mid i \in I\})(x', x)$ as the probability that event e will occur in the interval $[x', x]$ where x and x' are relative times in respect to the time l when state

s was entered, and given that s was reached by following a trace in $\{T_i \mid i \in I\}$. In other words,

$$Av\operatorname{Re}s^l(e,s,\{T_i \mid i \in I\})(x',x) =$$

$$Av\operatorname{Re}s^l(e,s,\{T_i \mid i \in I\})(x) - Av\operatorname{Re}s^l(e,s,\{T_i \mid i \in I\})(x')$$

3. $Av\operatorname{Re}s^l(e,s,T)$, where T is a trace that leads to state s, to be equal to $Av\operatorname{Re}s(e,s,\Theta)$ where Θ is the set of all possible regenerative traces that have T as a postfix. If Θ contains only one trace, then $Av\operatorname{Re}s^l(e,s,\Theta) = \operatorname{Re}s^l(e,s,\Theta)$.

As mentioned in the introduction, unless otherwise mentioned, we assume throughout this chapter that the GSMP has been running for a sufficiently long time. In that case, we assume that $l \to \infty$ and omit the suffix l from the notation of the residual time distributions.

Lemma 4.2. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, and let $s \in S$. Let T_1, T_2 be two traces from s_0 to s such that T_1 and T_2 have the same postfix T, where T is a regenerative trace, then $\operatorname{Re}s(e,s,T_1) = \operatorname{Re}s(e,s,T_2) = \operatorname{Re}s(e,s,T)$

∎

The next definition presents the single-AvRes set of traces, it is a set of traces Γ between two states s and s' for which $Av\operatorname{Re}s(e,s',\Gamma)$ has a single calculable value through the use of analytical means and this is the reason for its name.

Definition 4.8: Single-AvRes set of traces.

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP. Let $T = s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{n-1}} s_n$ be a path of G (recall that a path is a simple trace, i.e. a trace with no cycles in it), and we write $t_i = s_i \xrightarrow{e_i} s_{i+1}$. Then the *single-AvRes set of traces for path* T, written T^{AvR}, is defined as follows:

1. For $i \neq n$, we define E_i to be the nest of s_i for which $s_{i+1} \in E_i$, if any; if no such nest exists, E_i is assumed to be composed of the single state s_i.

2. Writing S_{E_i} for the set of states of E_i and $\mapsto_{E_i}$ for the set of traces in E_i, let M be the sub-GSMP whose states are $\bigcup_{i=1}^{n}(S_{E_i})$, and whose transitions are $\bigcup_{1}^{n-1} t_i \cup \bigcup_{1}^{n}(\mapsto_{E_i})$, then the set of traces in M from s_1 to s_n is called a *single-AvRes* set for path T and is denoted by T^{AvR}.

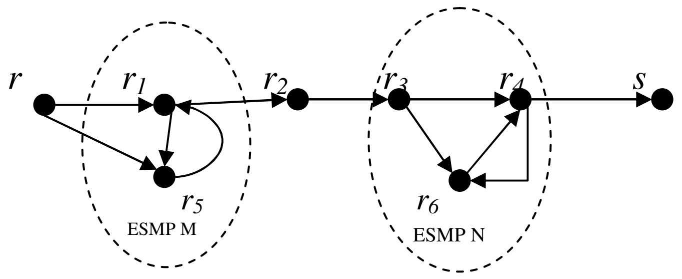


Figure 8. An example of a single-AvRes-set

Figure 8 shows an example of a *single-AvRes*-set of traces for path $T = r \rightarrow r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow s$. The set of all traces in the figure from state r to state s is a *single-AvRes*-set. The same set of traces is also the *single-AvRes*-set for path $T = r \rightarrow r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_6 \rightarrow r_4 \rightarrow s$.

The following definition presents a structural bisimulation equivalence between GSMPs, which was introduced in [29]. This bisimulation will be needed in the proofs of Section 4.4. It will become obvious when we formally define the s-simulation that structural bisimulation implies s-simulation, while the reverse is not true.

Definition 4.9: Structural bisimulation

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}, F', A', \mapsto', K')$ be two GSMPs.

We say that G is *structurally bisimilar* to G' if there exists a relation $R \subseteq S \times S'$, such that $(s_0, s'_0) \in R$ and whenever $(s, r) \in R$ the following conditions hold:

1. for all $s' \in S$, $s \xrightarrow{e} s'$ there exists a state $r' \in S'$ such that $r \xrightarrow{e} r'$, and $(s', r') \in R$
2. for all $r' \in S'$, $r \xrightarrow{e} r'$ there exists a state $s' \in S$ such that $s \xrightarrow{e} s'$ and $(s', r') \in R$;
3. $K(s) = K(r)$

Figure 9 shows a GSMP, note that states 0-0, 3-3, 4-5 and 5-4 are regenerative (note that transition *d* is immediate). We will later see that the GSMP of Figure 3 is structurally bisimilar to the GSMP shown in Figure 9. The relation that establishes the simulation is: every state *i-j* of Figure 9 corresponds to state *j* in Figure 3.

It can be easily shown that if two GSMPs $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ are structurally bisimilar. And if $\Delta = \{\Delta_j, j \in J\}$ is the partition of states S' induced by the bissimulation relation, and if $R(s) = \Delta_j = \{r_1, ..., r_n\}$ for some $s \in S$ and $j \in J$, then $P^G(s(x)/s_0(0)) = \sum_{i=0}^{n} P^{G'}(r_i(x)/s'_0(0))$. For more information on structural bisimulation the reader is referred to [31],[32],[30].

4.3.2. Properties of NRGSMP.

Theorem 4.2. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an NRGSMP, and $s \in S$. Let $\Gamma_s = \{T_1, T_2, ..., T_n\}$ be the set of all regenerative paths leading to s that contain exactly one regenerative state (the starting state). Then:

1. the set $\Gamma_s^{AvR} = T_1^{AvR} \cup T_2^{AvR} \cup ... \cup T_n^{AvR}$ has the property: $Av\mathrm{Re}s(e, s, \Gamma_s^{AvR}) = Av\mathrm{Re}s(e, s)$.

2. For all $i, j \in \{1, ..., n\}$ either $T_i^{AvR} = T_j^{AvR}$ or $T_i^{AvR} \cap T_j^{AvR} = \phi$

Proof.

- The proof of point 1 is based on the fact that if T is any path from s_0 to s, then T has a postfix T_i from the set $\Gamma_s = \{T_1, T_2, ..., T_n\}$, and hence $Av\mathrm{Re}s(e, s, T^{AvR}) = Av\mathrm{Re}s(e, s, T_i^{AvR})$.

- Point 2: Assume that there exists a trace $T \in T_i^{AvR} \cap T_j^{AvR}$, then since $T \in T_i^{AvR}$ we deduce from Definition 4.8 that $T^{AvR} = T_i^{AvR}$, and similarly, $T^{AvR} = T_j^{AvR}$.

∎

In the next subsection, we will introduce the hidden Markov regenerative process. It is needed as an intermediate step in the transformation of an NRGSMP into an SMP.

4.3.3. HMRP Definition and Properties

A hidden Markov regenerative process (HMRP) is given this name because it can be transformed into a Markov regenerative process: As we will see in this section, for every state *s* in a HMRP, if Λ_s is the set of all regenerative traces leading to *s* that contain only one regenerative state, then Λ_s is a single-*AvRes* set, and as a consequence, we will be able to calculate $Av\,\mathrm{Re}\,s(e,s)$ for all $e \in A(s)$ (refer to Section 4.2). Then given $Av\,\mathrm{Re}\,s(e,s)$ for all $e \in A(s)$, we can transform the state s into a regenerative one by having all events $e \in A(s)$ initialized in state s according to the distribution $F_s(e)(x) = Av\,\mathrm{Re}\,s(e,s)(x)$.

In this sub-section, we will present the properties of an HMRP, and we will start with their definition. Then in the next section we will present the algorithm to transform an NRGSMP into an HMRP, and then an HMRP into an SMP.

Definition 4.10. HMRP, defining subtrace of a state

A *hidden Markov regenerative process*, HMRP, is an NRGSMP such that, for every state s, if T is a regenerative path from some state r to s, such that the only regenerative state of T is r, then all traces leading to s have a trace in T^{AvR} as a postfix. The path T is called a *defining subtrace for state* s.

Figure 9 shows a HMRP, note that states 0-0, 3-3, 4-5 and 5-4 are regenerative. The relation that establishes the simulation is: every state *i-j* of Figure 9 corresponds to state *j* in Figure 3.

Figure 10 shows another example of an HMRP. The HMRP represents the failure and repair for a machine with two processors. One processor can be working at a time, the other remains idle until the working processor fails (modeled by f). When one processor fails it undergoes regular repair (modeled by r). If both processors fail, the machine undergoes major repair (modeled by m). Every state is annotated with two letters (representing the two processors) from the set {W,I,F} where W stands for working , I stands for idle and F stands for failed (refer to Figure 10). Note that the HMRP depicted in Figure 10 may not satisfy the enabling restriction because f and r, both, may not be exponentially distributed.

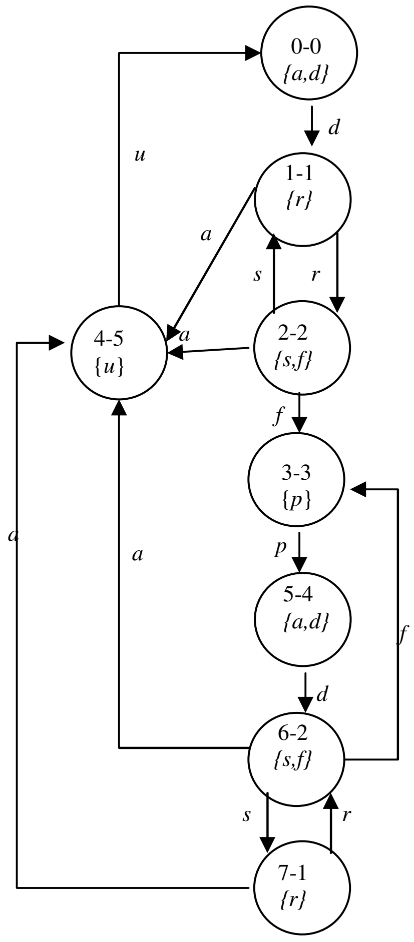


Figure 9. An HMRP

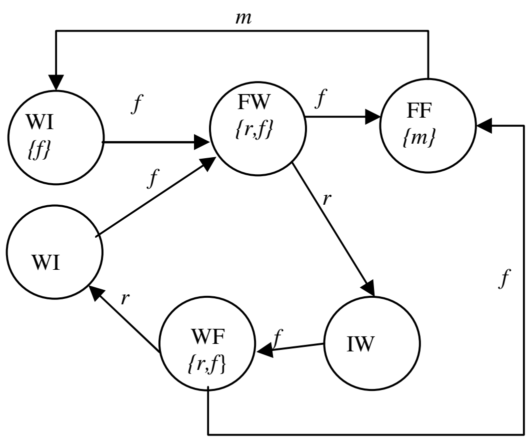


Figure 10. Example of an HMRP.

We will next examine the properties of an HMRP in detail.

Theorem 4.3. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an HMRP. Let T and T' be two regenerative paths leading to a state s. If r and r' are the last regenerative states that T and T' pass through respectively, then $r = r'$.

Proof. Let T_r be the postfix of trace T starting from r, we know from Definition 4.10 that trace T' has a trace in T_r^{AvR} as a postfix. Moreover all states belonging to the set of traces T_r^{AvR} aside from r are non-regenerative (see Definition 4.8), hence the last regenerative state in T' is r.

∎

Corollary 4.1. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an HMRP. Let T be any regenerative path leading to a state s. Let T' be the postfix of T, such that T' has only one regenerative state, then:

- $Av\mathrm{Re}\,s(e, s, T^{AvR}) = Av\mathrm{Re}\,s(e, s, T'^{AvR})$ and
- $Av\mathrm{Re}\,s(e, s) = Av\mathrm{Re}\,s(e, s, T^{AvR})$

Proof.

- The first point is due to the fact that a regenerative trace is memoryless.
- Since every path leading to s has a postfix in T'^{AvR}, and since all traces in T'^{AvR} are regenerative then $Av\mathrm{Re}\,s(e, s) = Av\mathrm{Re}\,s(e, s, T'^{AvR})$.

∎

Corollary 4.2. Let $\{T_1, ..., T_n\}$ be the set of defining traces for state s, then we have that:

$T_i^{AvR} = T_j^{AvR}$ for any $i, j \in \{1,...,n\}$

Proof. This corollary is a direct consequence of Theorem 4.3

∎

Theorem 4.4. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an HMRP. Let $s \in S$, and let T_s be a defining trace for state s such that T_s travels from state r_1^1 to state s. Then the set of traces T_s^{AvR} is of the form: $T_1(N_1(r_1^{n_1})^*)_{r_1^{n_1}}^{r_2^1} T_2(N_2(r_2^{n_2})^*)_{r_2^{n_2}}^{r_3^1} ... T_m(N_m(r_m^{n_m})^*)_{r_m^{n_m}}^{s}$ where $T_s = T_1T'_1T_2T'_2...T_{m-1}T'_{m-1}T_mT'_m$, $T_i = r_i^1 \to r_i^2 \to ... \to r_i^{n_i}$, and $N_i(r_i^{n_i})$ is the nest of $r_i^{n_i}$ that contains all states belonging to T'_i, and $(N_i(r_i^{n_i})^*)_{r_i^{n_i}}^{r_{i+1}^1}$ are the set of all traces in $N_i(r_i^{n_i})$ that start with state $r_i^{n_i}$ and end with state r_{i+1}^1.

Proof. The proof is straightforward from Definition 4.8.

∎

Corollary 4.3: Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an HMRP, let $s \in S$ be a state in a nest N_m of r, let T be a path from r to s,

1. Every trace from s_0 to s passes through r

2. If r is regenerative, then T is a defining subtrace for state s

Proof. To understand Point 1, we note that, from Theorem 4.4, every regenerative trace leading to s has a postfix of the form $T_m(N_m(r)^*)_r^s$. Point 2 is a direct consequence of Point 1.

∎

Theorem 4.5. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an HMRP, let $s \in S$ and let $T_s = s_1 \to s_2 ... \to s_m = s$ be a defining subtrace for s. We note that:

$T_{s_i} = s_1 \to s_2 ... \to s_i$ is a defining subtrace for s_i.

Now define T_i as follows: $T_s = T_{s_i} T_i$ and let $G' = (S, s_0, \mathrm{E}, F', A, \mapsto, K')$ be the HMRP where F' and K' are defined as follows: for all $r \in S - \{s_i\} : K'(r) = K(r)$ and for all $e \in K(r) : F'_r(e) = F_r(e)$, $K'(s_i) = A(s_i)$ and $F_{s_i}(e) = Av\mathrm{Re}\,s(e, s_i T_{s_i}^{AvR})$ for all $e \in A(s_i) - K(s_i)$.

Then:

2- The defining trace for s in G' is T_i, and

3- $Av\mathrm{Re}\,s^{G}(e, s, T_s^{AvR}) = Av\mathrm{Re}\,s^{G'}(e, s, T_i^{AvR})$

Proof. The proof is straightforward.

∎

So, to sum up, given a state s in G, the average residual time for every active event e in state s can be obtained from a defining trace T_s for state s: $Av\mathrm{Re}\,s(e, s, T_s^{AvR})$. Hence, if we can find a way to calculate $Av\mathrm{Re}\,s(e, s, T_s^{AvR})$ were T_s^{AvR} has the form specified in Theorem 4.4 then s could be transformed into a regenerative state. Theorem 4.5 tells us that the residual time distributions for the events of one state can be calculated from the residual times of the events of another state. In sub-Section 4.3.4, we will present the method used to calculate $Av\mathrm{Re}\,s(e, s, T_s^{AvR})$.

4.3.4 Properties of Average Residual Times

In this section, we do not assume that the GSMP is in steady state. The purpose is to present some properties of the $Av\mathrm{Re}s^l$. We will first prove that if t_1 is a transition in a GSMP G from states s_1 to s_2, then $Av\mathrm{Re}s^l(e, s_2, t_1)(x)$, where l is the time state s was entered, can be calculated from the average residual time distributions of events in s_1. Then we will prove that, given a nest N of a state r_1, then for any $s \in N$, $Av\mathrm{Re}s^l(e, s, N(r_1)^{s}_{r_1}*)$ can be calculated from the residual times for events in r_1 and from the steady state probability of state s relative to an extended SMP N' which is obtained from N with some additional states and transitions. In Section 4.4, we will explain how we can calculate $Av\mathrm{Re}s^l(e, s)(x)$ for any state s in an HMRP G using the two results described in this section.

First, we present some notations that will be used throughout this section:

- $P(A \mid T)$ is the probability that event A occurs given that trace T will occur next.
- $P(T \mid A)$ is the probability that trace T will be followed, given that event A had taken place.
- $\downarrow s$ means that we are in state s
- $e(s)[x', x'+x]$ means that event e occurs in the interval $[x', x'+x]$ counting from the time we entered state s.

$\blacksquare$

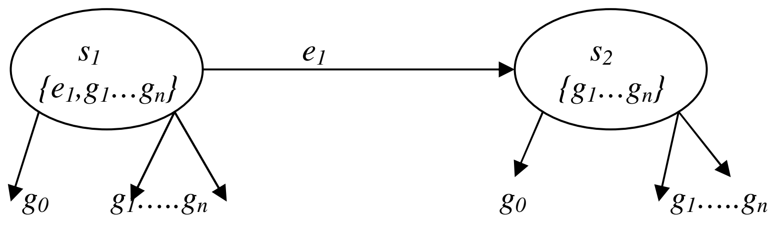


Figure 11. Two consecutive states

In the next theorem, given two consecutive states s_1 and s_2 and an event g_0 that is active in both states (Figure 11), we will show how to calculate the distribution of the residual lifetime of g_0 from the distribution of the residual lifetimes of the events in $A(s_1)$. To explain the result informally, note that if we are in state s_2 after following trace $s_1 \xrightarrow{e_1} s_2$, then g_0 had been active for $\varsigma(s_1)$ time units since reaching state s_1, so the lifetime of g_0 has been shortened by $\varsigma(s_1)$ time units. (Note that $\varsigma(s_1) \in [0,l]$ because there is a limit l on the time that the process has been running for). In fact, we will show below that the probability that g_0 will expire within x time units from entering state s_2 given that trace $s_1 \xrightarrow{e_1} s_2$ had just occurred is:

$Av\operatorname{Re}s^l(g_0, s_2, s_1 \xrightarrow{e} s_2)(x) = \int_0^l \frac{P(g_0(s_1)[x', x'+x] \wedge \varsigma(s_1) = x' \mid s_1 \xrightarrow{e} s_2)}{dx'} dx'$ i.e. it is equal to the probability that $\varsigma(s_1) = x'$ and that g_0 expires from state s_1 in the interval $[x', x + x']$, where $x' \in [0,l]$.

Refer to Figure 11 for a better understanding of the theorem.

Theorem 4.6. Let t_1 be a transition in a GSMP G from states s_1 to s_2 with $t_1 = s_1 \xrightarrow{e_1} s_2$. Assume that , and $A(s_1) = \{g_0, e_1, g_1, \ldots, g_n\}$, then

$Av\operatorname{Re}s^l(g_0, s_2, t_1)(x) =$

$$\int_0^l \frac{Av\operatorname{Re}s^l(g_0,s_1)(x+x') - Av\operatorname{Re}s^l(g_0,s_1)(x')}{1-Av\operatorname{Re}s^l(g_0,s_1)(x')}$$

$$\frac{\frac{dAv\operatorname{Re}s^l(e_1,s_1)(x')}{dx'}\prod_{i=0}^{n}(1-Av\operatorname{Re}s^l(g_i,s_1)(x'))}{\int_0^l \frac{dAv\operatorname{Re}s^l(e_1,s_1)(x'')}{dx'}\prod_{i=0}^{n}(1-Av\operatorname{Re}s^l(g_i,s_1)(x''))dx''}dx'$$

Proof. In the proof below, we assume that the process is in state s_1.

$$Av\operatorname{Re}s^l(g_0,s_2,t_1)(x) = P(g_0(s_2)[0,x] \mid t_1) = \int_0^l \frac{P(\varsigma(s_1)=x') \wedge g_0(s_2)[0,x] \mid t_1)}{dx'}dx'$$

$$= \int_0^l \frac{P(\varsigma(s_1)=x') \wedge g_0(s_1)[x',x'+x] \mid t_1)}{dx'}dx'$$

Applying the formulae $P(A \wedge B \mid C) = P(A \mid B \wedge C)P(B \mid C)$ we get

$$\int_0^l P(g_0(s_1)[x',x'+x] \mid t_1 \wedge \varsigma(s_1)=x')\frac{P(\varsigma(s_1)=x' \mid t_1)}{dx'}dx'$$

Applying the formulae $P(A \mid B) = \dfrac{P(A \wedge B)}{P(B)}$, we get:

$$\int_0^l \frac{P(g_0(s_1)[x',x'+x] \mid t_1)}{P(\varsigma(s_1)=x') \mid t_1)}\frac{P(\varsigma(s_1)=x' \mid t_1)}{dx'}dx'$$

$$= \int_0^l \frac{Av\operatorname{Re}s^l(g_0,s_1)(x+x') - Av\operatorname{Re}s^l(g_0,s_1)(x')}{1-Av\operatorname{Re}s^l(g_0,s_1)(x')}\ \frac{P(\varsigma(s_1)=x' \mid t_1)}{dx'}dx'$$

Now applying the formulae $P(A \mid B) = \dfrac{P(A \wedge B)}{P(B)}$ on the second term we get:

$$\int_0^l \frac{Av\operatorname{Re}s^l(g_0,s_1)(x+x') - Av\operatorname{Re}s^l(g_0,s_1)(x')}{1-Av\operatorname{Re}s^l(g_0,s_1)(x')}\ \frac{P(\varsigma(s_1)=x' \wedge t_1 \mid\downarrow s_1)}{P(t_1 \mid\downarrow s_1)dx'}dx'$$

$$= \int_0^l \frac{Av\operatorname{Re}s^l(g_0, s_1)(x+x') - Av\operatorname{Re}s^l(g_0, s_1)(x')}{1 - Av\operatorname{Re}s^l(g_0, s_1)(x')}$$

$$\frac{\frac{dAv\operatorname{Re}s^l(e_1, s_1)(x')}{dx'} \prod_{i=0}^{n} (1 - Av\operatorname{Re}s^l(g_i, s_1)(x'))}{\int_0^l \frac{dAv\operatorname{Re}s^l(e_1, s_1)(x'')}{dx'} \prod_{i=0}^{n} (1 - Av\operatorname{Re}s^l(g_i, s_1)(x''))dx''} dx'$$

∎

Now, let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, assume that for all $s \in S$, if $f \in A(s)$ and f has an exponentially distributed lifetime, then $f \in K(s)$. Now, let $N(r_1)$ be a nest for state r_1 in G, let S' be the state space of $N(r_1)$, and let $s \in S' - \{r_1\}$. Then for all $e \in A(s) - K(s)$, we want to calculate $Av\operatorname{Re}s^l(e, s, (N(r_1)^*)^s_{r_1})$. Note that since $e \in A(s) - K(s)$ then $e \in A(r_1)$ and $e \notin \bigcup_{r \in S' - \{r_1\}} K(r)$. If r_1 is regenerative, i.e. if $e \in K(r_1)$, then once we reach state s, event e would be active from the time we entered r_1. Hence, to calculate $Av\operatorname{Re}s(e, s, (N(r_1)^*)^s_{r_1})$, we need to subtract from the distribution of event e the amount of time taken to reach s from r_1. In other words:

$$Av\operatorname{Re}s^l(e, s, (N(r_1)^*)^s_{r_1}) = \int_0^l \frac{P(e(s)[x', x'+x] \wedge \varsigma(r_1 \to s) = x' \mid N(r_1)^*)^s_{r_1})}{dx'} dx'$$

Where $\varsigma(r_1 \to s) = x'$ represents the fact that state s is entered after x' time units of entering r_1.

Formally, this can be done as follows:

Let $\{e'_1, e'_2, \ldots e'_m\} = \bigcup_{r \in S' - \{r_1\}} K(r)$ and let $\{e, e_1, e_2, \ldots, e_l\} = \bigcup_{r \in S' - \{r_1\}} A(r) - \bigcup_{r \in S' - \{r_1\}} K(r)$, i.e. $\{e, e_1, e_2, \ldots e_v\} \in A(r_1)$. We can assume without loss of generality that $K(r) \cap K(r') = \phi$ for all $r \neq r' \in S' - \{r_1\}$. Let g_i be the average residual distribution for e_i in r_1 where $i \in \{1, \ldots, v\}$

and let f_i be the time distributions for e'_i in the state where they are initialized, for $i \in \{1,...,m\}$.

We need to find $Av\operatorname{Re}s^l(e,s,(N(r_1)^*)^s_{r_1}))$, where $(N(r_1)^*)^s_{r_1})$ represents all traces in $N(r_1)$ starting with r_1 and ending in s. We recall from the definition of ESMPs that since $e \in A(s) - K(s)$ then $e \notin K(r)$ for all $r \in S' - \{r_1\}$. Now, assume that the average residual for all events in state r_1 are known, i.e. the distributions g_i where $i \in \{1,...,v\}$ are known, in other words we assume that r_1 was transformed into a regenerative state, then if we entered state s after x' time units of entering r_1, then $Av\operatorname{Re}s^l(e,s,(N(r_1)^*)^s_{r_1}))(x)$ (which is the probability that event e occurs in s within x time units of entering s), is defined as:

$$Av\operatorname{Re}s^l(e,s,(N(r_1)^*)^s_{r_1}))(x) = \int_0^l \frac{P(\varsigma(r_1 \to s) = x' \mid N(r_1)^{*s}_{r_1})}{dx'} P(e)dx' \qquad (1)$$

where $\varsigma(r_1 \to s) = x'$ represents the fact that state s is entered after x' time units of entering r_1, and $P(e)$ is the probability that e occurs between x' and $x + x'$ time units since entering r_1 given that e does not occur before x'. So $P(e) = \dfrac{Av\operatorname{Re}s^l(e,r_1)(x'+x) - Av\operatorname{Re}s^l(e,r_1)(x')}{1 - Av\operatorname{Re}s^l(e,r_1)(x')}$

And if we apply the rule $P(A \wedge B \mid C) = \dfrac{P(A \wedge C \mid B)}{P(C \mid B)}$ to the first term of Equation (1), we get:

$$\frac{P(\varsigma(r_1 \to s) = x' \mid N(r_1)^{*s}_{r_1})}{dx'} =$$

$$\frac{P((\varsigma(r_1 \to s) = x') \wedge N(r_1)^{*s}_{r_1} \mid \downarrow r_1)}{dx'} \frac{1}{P(N(r_1)^{*s}_{r_1} \mid \downarrow r_1)} \qquad (2)$$

Before we proceed, let $\overrightarrow{N(r_1)}$ be the semi-Markov process formed from $N(r_1)$ with the addition of all missing transitions out of any state in $N(r_1)$ that is governed by an event from $\{e'_1, e'_2, ..., e'_m\}$ and the states they lead to, i.e. all transitions that lead you outside $N(r_1)$ and that are governed by events initialized inside $N(r_1)$. The states added to $\overrightarrow{N(r_1)}$ are made into absorbing states with respect to $\overrightarrow{N(r_1)}$, so the probability of being in one of these states equals the probability of getting out of $N(r_1)$ on a transition from the set $\{e'_1, e'_2, ...e'_m\}$. Denote by S_a the set of absorbing states of $\overrightarrow{N(r_1)}$ that do not belong to $N(r_1)$.

Now, denote by

- $E_1\,[0, x']$ the fact that events $\{e_1, e_2, ...e_v\}$ do not occur between $[0, x']$ since entering r_1, then $P(E_1[0, x']) = \prod_{i=1}^{v}(1 - g_i(x'))$ and

- $E_2\,[0, x']$ the fact that no state in the set S_a is visited in the interval $[0, x']$ since entering r_1 given fact E_1. $P(E_2[0, x'])$ is a function of the total time spent in a state in the set S_a in the interval $[0, x']$ relative to the SMP $\overrightarrow{N(r_1)}$. $P(E_2[0, x']) = 1 - \frac{1}{x'} \sum_{r \in S_a} L_r^{\overrightarrow{N(r_1)}}(x')$.

- $E_3[0, x']$ the fact that we are in state s after x' time units since entering r_1, given that $E_1[0, x']$, and $E_2[0, x']$. Hence $P(E_3[0, x'])$ is the transient state probability for state s relative to SMP $N(r_1) =. \pi_s^{N(r_1)}(x')$

Now, the left term of Equation [3],

$\frac{P(\varsigma(r_1 \to s) = x' \wedge N(r_1) *_{r_1}^{s} | \downarrow r_1)}{dx'} = P($ {events $\{e_1, e_2, ...e_v\}$ do not occur between $[0, x']$ relative to the time r_1 was entered} $\wedge$ {no state from the set S_a is visited in the interval $[0, x']$ } $\wedge$ $\{s \downarrow (x' + l) \,|\, r_1(l))\}$ $\frac{1}{dx'}$

Now applying the formulae: $P(A \wedge B \wedge C \mid D) = P(A \mid D)P(B \mid A \wedge D)P(C \mid A \wedge B \wedge D)$, we get:

$$\frac{P(\varsigma(r_1 \to s) = x' \wedge N(r_1) *^{s}_{r_1} \mid \downarrow r_1)}{dx'}$$

$= P($ {events $\{e_1, e_2, ... e_v\}$ do not occur between $[0, x']$ relative to the time r_1 was entered} $\wedge$ {no state from the set S_a is visited in the interval $[0, x']$} $\wedge s \downarrow (x'+l) \mid r_1(l))\ \dfrac{1}{dx'}$

$$= P(E_1[0,x'])P(E_2[0,x'])\frac{dP(E_3[0,x'])}{dx'}$$

$$= \prod_{i=1}^{v}(1-[g_i(x')-g_i(0)])\,[1-\frac{1}{x'}\sum_{r\in S_a} L_r^{\overrightarrow{N(r_1)}}(x')\,]\,\frac{d\pi_s^{N(r_1)}(x')}{dx'}$$

And the bottom of the right term of Equation (2) is

$$P(N(r_1) *^{s}_{r_1} \mid \downarrow r_1) = \int_0^l \frac{P(\varsigma(r_1 \to s) = x' \wedge N(r_1) *^{s}_{r_1} \mid \downarrow r_1)}{dx'} dx'$$

$$= \int_0^l \ \{ \prod_{i=1}^{v}(1-[g_i(x')-g_i(0)])\,[1-\frac{1}{x'}\sum_{r\in S_a} L_r^{\overrightarrow{N(r_1)}}(x')\,]\,\frac{d\pi_s^{N(r_1)}(x')}{dx'} \ \}\, dx'$$

From the above arguments, we conclude the following Theorem:

Theorem 4.7. Let G be a GSMP, let $N(r_1)$ be a nest for r_1 in G. We denote by S the set of states of $N(r_1)$, let $s \in S - \{r_1\}$, let $\{e_1, e_2, ... e_v\} = \bigcup_{r\in S-\{r_1\}} A(r) - \bigcup_{r\in S-\{r_1\}} K(r)$ and let $\{e'_1, ..., e'_m\} = \bigcup_{r\in S-\{r_1\}} K(r)$. Let g_i be the time distribution for e_i where $i \in \{1,...,v\}$ and let f_i be the time distributions for e'_i where $i \in \{1,...,m\}$. Let $\overrightarrow{N(r_1)}$ be the process formed from

$N(r_1)$ with the addition of all transitions governed by events $\{e'_1,...,e'_m\}$ and their corresponding next states. And by S_a set of absorbing states of $\overrightarrow{N(r_1)}$ that do not belong to $N(r_1)$.

Then for all $e \in A(s) - K(s)$:

$Av\,\mathrm{Re}\,s^l(e,s,(N(r_1)^*)^s_{r_1})\ (x) =$

$$\int_0^l \frac{\prod_{i=1}^{v}(1-g_i(x'))[1-\frac{1}{x'}\sum_{r\in S_a} L_r^{\overrightarrow{N(r_1)}}(x')]\frac{d\pi_s^{N(r_1)}(x')}{dx'}}{\int_0^l \prod_{i=1}^{v}(1-g_i(x''))[1-\frac{1}{x''}\sum_{r\in S_a} L_r^{\overrightarrow{N(r_1)}}(x'')]\frac{d\pi_s^{N(r_1)}(x'')}{dx''}dx''} \frac{Av\,\mathrm{Re}\,s^l(e,r_1)(x'+x) - Av\,\mathrm{Re}\,s^l(e,r_1)(x')}{1-Av\,\mathrm{Re}\,s^l(e,r_1)(x')}dx'$$

where $\pi_s^{N(r_1)}(x')$ is the transient state probability for state s relative to the SMP $N(r_1)$ and $\sum_{r\in S_a} L_r^{\overrightarrow{N(r_1)}}(x')$ is the total time spent in the states of the set S_a in the interval $[0,x']$ relative to the SMP $\overrightarrow{N(r_1)}$.

∎

Note that both theorems apply to the steady state case by taking $l \to \infty$.

4.3.5. Transient and Steady-State Simulations: Definitions and Properties.

In this section, we will present the definition of s-simulation, and we will prove that if an NRGSMP s-simulates another, then the steady state probabilities of the latter can be deduced from the former. As a generalization of s-simulation, we will present another simulation called transient state simulation or t-simulation as a separate definition, then we will prove that if an NRGSMP t-simulates another, then the transient state probability of the latter can be deduced from the former. Properties of both simulations will be given in two separate theorems.

Definition 4.11: Steady-state simulation

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}, F', A', \mapsto', K')$ be two NRGSMPs.

We say that G' *steady-state simulates* (or *s-simulates)* G if there exists a partition $\Delta = \{\Delta_j, j \in 1,...,|S|\}$ of the states in S', and a bijection $R : S \to \Delta$ such that $\{s'_0\} \in \Delta$ and $R(s_0) = \{s'_0\}$ and for all regenerative states $r \in S$, we have that $R(s) = \{s'\}$ for some state $s' \in S'$. Moreover, for every state $s_1 \in S$ if $R(s_1) = \Delta_1 = \{s'_1, ..., s'_{n_1}\}$ then:

1. If $s_1 \xrightarrow{e} s_2 \in \mapsto$, then for each $i \in \{1,...,n_1\}$ there exists a state $s'^2_i \in R(s_2)$ and a transition $s'_i \xrightarrow{e} s'^2_i \in \mapsto'$, such that if $e \in K(s_1)$ then $e \in K'(s'_i)$ and $F_{s_1}(e) = F'_{s'_i}(e)$, and inversely,
2. For each $i \in \{1,...,n_1\}$ if $s'_i \xrightarrow{e} s'^2_i \in \mapsto'$ then there exists a state s_2 with $s'^2_i \in R(s_2)$ and a transition $s_1 \xrightarrow{e} s_2 \in \mapsto$.
3. For $i \in \{1,...,n_1\}$, let $\Gamma'_{s'_i}$ be the set of all single regenerative traces leading to s'_i in G', then we have that $Av\mathrm{Re}s(e, s_1, R^{-1}(\Gamma'_{s'_i})) = Av\mathrm{Re}s(e, s'_i, \Gamma'_{s'_i})$ for all $e \in A(s_1)$.

To illustrate the above definition, assume that G' s-simulates G through a partition Δ. Let $G'/\Delta = \{\Delta, \{s'_0\}, E, F'', A'', \mapsto'', K'')$ be the GSMP such that, for all $\Delta_1, \Delta_2 \in \Delta$: $\Delta_1 \xrightarrow{e} \Delta_2$ iff for all $s_1 \in \Delta_1$ and for all $s_2 \in \Delta_2$ we have $s_1 \xrightarrow{e} s_2$, then from a functional point of view (i.e. if we neglect the timing constraints), G and G'/Δ would be identical. On the other hand, we have from Points 1 and 2 that for every $s \in S$ if $s' \in R(s)$ and if $\Gamma'_{s'_i}$ is the set of all single regenerative traces leading to s' then $K(s) \subseteq K(s')$, and from Point 3 that if $e \in K(s') - K(s)$, then e has the same average residual distribution in both states s and s' given that we reached s through a trace from the set $R^{-1}(\Gamma'_{s'_i})$.

Note that if $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ s-simulates $G' = (S', s'_0, \mathrm{E}, F', A', \mapsto', K')$ and if $K'(r) = K(R^{-1}(r))$ for all $r \in S'$, then G is *structurally bisimilar* to G'

Now we present some properties of the steady-state simulation, among which the property which states that if a GSMP steady-state simulates another one, then the steady state probabilities of the latter could be deduced from the steady state probabilities of the former.

Theorem 4.8. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be two GSMPs such that G' s-simulates G. Let $\Delta = \{\Delta_j, j \in J\}$ be the partition of states S' and $R : S \to \Delta$ be the correspondence that establishes the steady-state simulation. Now let $s \in S$, and assume that $R(s) = \{r_1, ..., r_n\} \in \Delta$, and let Γ'_{r_i} be the set of all single regenerative traces leading to r_i in G', and assume that $\Gamma_i = R^{-1}(\Gamma'_{r_i})$ then we have that:

1. $\varsigma^G(s \mid \Gamma_i) = \varsigma^{G'}(r_i)$ where $\varsigma^G(s \mid \Gamma_i)$ is the distribution of the sojourn time in state s at equilibrium given that a trace from Γ_i was followed.

2. If t'_i is a transition out of state r_i in G' and if $t = R^{-1}(t'_i)$, then $\pi^G(t \mid \Gamma_i) = \pi^{G'}(t'_i)$ where $\pi^G(t \mid \Gamma_i)$ is the probability at equilibrium that transition t occurs from state s given that a trace from Γ_i was followed to reach state s.

3. $\pi^G(s) = \sum_{i=1}^{n} \pi^{G'}(r_i)$.

Proof. We prove the four point one by one:

1. To prove this point, it is enough to observe that $\varsigma^{G}(s \mid \Gamma_i)$ is a function of the residual lifetimes of the active events in state s knowing that a trace from Γ_i was followed to reach s.

2. Similarly, to prove this point, it is enough to observe that $\pi^{G}(t \mid \Gamma_i)$ is a function of the residual lifetimes of the active events in state s knowing that a trace from Γ_i was followed to reach s.

3. To prove this point, let $G" = (S', s'_0, \mathrm{E}, F", A', \mapsto', K")$ be the GSMP that is isomorphic to G' from a functional point of view, with $K"(r) = K(R^{-1}(r))$ and $F"_r(e) = F_{R^{-1}(r)}(e)$. In other words, G and $G"$ are structurally bisimilar. Then it is enough to prove that

   a. $\pi^{G}(s) = \sum_{i=1}^{n} \pi^{G"}(r_i)$ and

   b. $\pi^{G"}(r_i) = \pi^{G'}(r_i)$

   Property (a) is a direct consequence of structural bissimulation.

   Property (b): Let $\{v_1, ..., v_m\}$ be the set of all states from which r_i is directly accessible as follows: $v_j \xrightarrow{e_j} r_i$ for all $j \in \{1, ..., m\}$. Then, the probability of being in state r_i at time x is equal to the probability of having entered one of the v_j at time $x - x' - x"$ where $x' + x"$ is between 0 and x, and then moving to state r_i after spending $x"$ time units in v_j .and staying in r_i for at least x' time units, hence, in both GSMPs:

   $P(r_i(x) \mid s'_0(0) =$

$$\int_0^x \int_0^{x'} \sum_{j=1}^{m} \frac{dP(v_j(x-x'-x")\,|\,s'_0(0)}{dx'} \frac{d(Av\operatorname{Re}s^{x-x'-x"}(e_j,v_j,x"))}{dx} \prod_{e\in A(v_j)-\{e_j\}} [1-(Av\operatorname{Re}s^{x-x'-x"}(e,v_j,x"))](\zeta^{x-x'}(r_i)\geq x')dx"dx'$$

Where $\zeta^{x-x'}(r_i)$ is the soujourn time in state r_i given that state r_i was entered at time $x-x'$, hence

$$\lim_{x\to\infty} P(r_i(x)\,|\,s'_0(0) = \pi(r_i) =$$

$$\int_0^\infty \int_0^{x'} \sum_{j=1}^{m} \lim_{x\to\infty}\left(\frac{dP(v_j(x-x'-x")\,|\,s'_0(0)}{dx'}\right) \frac{d(Av\operatorname{Re}s(e_j,v_j,x"))}{dx} \prod_{e\in A(v_j)-\{e_j\}} [1-(Av\operatorname{Re}s(e,v_j,x"))](\zeta(r_i)\geq x')dx"dx'$$

But the average residual times are the same in both GSMPs, hence the steady state probabilities of consecutive states are related by the same formulae in both GSMPs. And hence they both have the same SSP.

∎

In what follows, we will generalize s-simulation by removing the assumption that the GSMP is in steady state. Then we will prove that if G' t-simulates G, then the transient state probability of G could be obtained from that of G'. The reason for the generalization is only to show how the equivalence generalizes to transient state, and the nice properties that one gets from t-simulation; but this is not relevant to the rest of the thesis, as only steady state properties will be considered.

Definition 4.11(2): Transient state simulation

Let $G=(S,s_0,\mathrm{E},F,A,\mapsto,K)$ and $G'=(S',s'_0,\mathrm{E},F',A',\mapsto',K')$ be two NRGSMPs. We say that G' *transient-state simulates* (or *t-simulates)* G if

1. G' steady-state simulates G and,

2. if $\Delta = \{\Delta_j, s_j \in S\}$ and $R : S \rightarrow \Delta$ are the partition and the bijection that establish the steady-state simulation, then for any $s_1 \in S$, if $R(s_1) = \Delta_1 = \{s'_1, ..., s'_{n_1}\}$, and if $\Gamma'_{s'_i}$ is the set of all single regenerative traces leading to a state s'_i in G', then, given that we entered states s_1 and s'_i after l time units of the start of the running of the system, then we have that $Av\mathrm{Re}s^l(e, s_1, R^{-1}(\Gamma'_{s'_i})) = Av\mathrm{Re}s^l(e, s'_i, \Gamma'_{s'_i})$ for all $e \in A(s_1)$.

In the next theorem, we will present the properties of the t-simulation.

Theorem 4.8(2). Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be two GSMPs such that G' t-simulates G. Let $\Delta = \{\Delta_j, j \in J\}$ be the partition of states S' and $R : S \rightarrow \Delta$ be the correspondence that establishes the transient state simulation. For any $s \in S$, if $R(s) = \{r_1, ..., r_n\} \in \Delta$, then $P^G(s(x)/s_0(0)) = \sum_{i=0}^{n} P^{G'}(r_i(x)/s'_0(0))$.

Proof. The proof is done in two steps:

1. Let $G'' = (S', s'_0, \mathrm{E}', F'', A', \mapsto', K'')$ be the GSMP such that $K''(r) = K(R^{-1}(r))$ and $F''_r(e) = F_{R^{-1}(r)}(e)$ for all $e \in A'(r)$. In other words, G'' and G are structurally bisimilar, then we have that $P^G(s(x)/s_0(0)) = \sum_{i=0}^{n} P^{G''}(r_i(x)/s'_0(0))$. So it is enough to prove that for all r in S', $P^{G'}(r(x)/s'_0(0)) = P^{G''}(r(x)/s'_0(0))$. This will be done in the next step.

2. Let $\{v_1, ..., v_m\}$ be the set of all states from which r is directly accessible as follows $v_i \xrightarrow{e_i} r$ for all $i \in \{1, ..., m\}$. Then, the probability of entering state r at time x is equal to the probability of having entered one of the v'_i at time x' where x' is

between 0 and x, and then moving to state r after spending $x\text{-}x'$ time units in v'_i. Hence:

$$\frac{dP^{G'}(r(x) \mid s'_0(0)}{dx} =$$

$$\int_0^x \sum_{i=1}^m \frac{dP^{G'}(v_i(x') \mid s'_0(0)}{dx'} \frac{d(Av\operatorname{Re}s^{x'}(e_i, v_i, x-x'))^{G'}}{dx} \prod_{e\in A(v_i)-\{e_i\}} [1-(Av\operatorname{Re}s^{x'}(e, v_i, x-x'))^{G'}]dx'$$

(Note that $\prod_{e\in A(v_i)-\{e_i\}} [1-(Av\operatorname{Re}s^{x'}(e, v_i, x-x'))^{G'}]$ represent the fact that events other than e_i that are active in v_i, occur after e_i)

But $(Av\operatorname{Re}s^{x'}(e, v_i, x-x'))^{G'} = (Av\operatorname{Re}s^{x'}(e, v_i, x-x'))^{G''}$ for all $e \in A(v_i)$, hence they both have the same set of equations, and hence the same transient probability.

∎

Note that, if an NRGSMP was transformed into an HMRP, and if all the states of the HMRP were transformed into regenerative states by assigning them their average residual distributions as a function of l, In other words, if for every state s, and every event $e \in A(s)$, we set $F_s(e) = Av\operatorname{Re}s^l(e,s)(x)$ which is a function of the time state s was entered: l, then the HMRP becomes a non-homogeneous semi-Markov process . We will not go into the specifics of this transformation as we are focusing on the steady state case. However, for more information on non-homogeneous semi-Markov process, the reader is referred to [58],[59].

4.4. From NRGSMP to SMP

In this section, we present the algorithms that transform an NRGSMP into an SMP. In Subsection 4.4.1 we present the algorithm that transforms the NRGSMP into an HMRP, then in Subsection 4.4.2 we present the algorithm that transforms the obtained HMRP into an SMP. An application will then be presented in Subsection 4.4.3.

4.4.1. Algorithm 1: NRGSMP to HMRP

In this Section, we will present an algorithm that transforms a NRGSMP into an HMRP such that the HMRP is structurally bisimilar to the NRGSMP (Theorem 4.9). In what follows we will adopt the following two assumptions:

- The NRGSMP is connected (in other words every state is reachable from the starting state).
- We assume that the regenerative states are not part of any non-trivial ESMP. For simplifying the description of the algorithm, we will say in this section that each regenerative state is a *reg-ESMP*, that is, a trivial ESMP that consists of a single state, namely the regenerative state in question.

4.4.1.1. Overview

We will start first with the following definition:

Definition 4.12: Execution tree, leaf node

- An *execution tree* of a GSMP G is a tree (graph with no cycles) that characterizes all possible execution paths that could be followed during the execution of G. The nodes of the tree represent states in G, and the arcs (also called transitions) connecting the nodes represent transitions between states in G.

In an execution tree, the root node corresponds to the starting state, and the number of transitions out of a node is equal to the number of transitions out of the corresponding state. An execution tree could be infinite.

- A *leaf node* in an execution tree is a node with no outgoing transitions.

The algorithm that transforms an NRGSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ into an HMRP G' works by constructing part of a "special execution tree" for the NRGSMP. Each node of the tree actually consists of copies of one or more state of G, such that all states in the same node belong to a reachable part of an ESMP. The nodes of this tree therefore represent a sub-ESMPs (i.e. every node is composed of a set of states and a set of transitions linking these states). For each transition between nodes n_1 and n_2, the transition is actually an arc between a particular state of n_1 to a particular state of n_2. We develop the tree by going down the tree until a leaf node is a reg-ESMP of G (i.e. it consists the copy of a single regenerative state of G) such that, this state is already represented by another node within the tree so far developed. The scenario is the following:

1. The root node of the tree is a reg-ESMP which is a copy of the starting state of the NRGSMP. At this point, the only leaf node is the root node.

2. For every leaf node n, do the following:

   (a) If n is a reg-ESMP and this reg-ESPM is already represented by another node n' in the tree built so far, then we stop expanding this node (because the reg-ESMPs of G will be presented by another node in the HMRP), and nodes n' and n are considered equivalent, and will be merged at the end. Otherwise,

   (b) for each copy of state s in the ESMP of node n, we do the following:

   Let $\{N_1,\ldots,N_n\}$ be the set of all ESMP in G having s as the in-border. Let N'_i be a copy of the sub-ESMP of N_i formed from the states of N_i that are reachable from

s. Then we create n nodes $\{N'_1,...,N'_n\}$, add them to the tree by creating arcs out of state s of n to the proper states in the sub-ESMPs $\{N'_1,...,N'_n\}$.

Note that, if the ESMP that forms node n contains copies of two states, say *s* and *s'*, such that the same sub-ESMP is reachable from both states, then two different nodes are created for the same sub-ESMP, one accessible from *s*, and the other from *s'*.

3. The last step would be to merge the leaf nodes that satisfied condition (a) with their equivalent nodes. Equivalent nodes are represent the same reg-ESMP.

Before presenting the different functions used in the algorithm, we need the following Corollary:

Corollary 4.4. Let G be an NRGSMP, and let G' be the HMRP obtained by applying the algorithm described above. The finiteness of the HMRP G' is guaranteed from the properties of G.

Proof. Let H be the tree obtained from the above algorithm without the last step 3, i.e. the tree whose nodes are sub-ESMPs, and define a relation R' over the states of G' as follows: $sR's'$ if s and s' belong to the same node in the tree H, assume that R' partitions the states of G' into the sets: $\{S_1,...S_n\}$. Then the following facts are straightforward:

1. For all $i \in \{1,...,n\}$, let M_i be the set of all states in G that are associated with the states in S_i. Then $|M_i| = |S_i|$.

2. Let C be a regenerative cycle in G. Let B be any branch in H. Let $\{n_0,...,n_l\}$ be the set of nodes in B. Let Γ be the set of all possible paths: $T = T_0T_1...T_L$, where T_i. is a trace inside node n_i. Then C is represented at most once in any trace of Γ.

3. If G' is infinite, then there should exist a state s of G that is represented infinitely many times in G'. Now, we know that G' has a finite branching factor (number of

states accessible from a given state through one transition). Hence there exists a **path** T in G' containing infinitely many states that represent s, say $T = T_1 r_1 T_2 r_2 ....$ where all r_is are associated with state s. Let m be the number of REG cycles that s belongs to in G, then from point (2) above, we conclude that each cycle is represented at most once in trace T. That means that there exists a sub-trace of T: $T' = r_1 T_2 r_2$, where T' represents the execution of an NSM cycle C of G. But states of an NSM cycle belong to the same ESMP, i.e. $r_1 R' r_2$ should be and. $r_1 = r_2$.

∎

4.4.1.2. Algorithm 1

The algorithm is composed of four major and six minor functions.

Major Functions:

a. GET-ESMP: This function finds all ESMPs belonging to the NRGSMP, and identifies the in-borders for each ESMP.

b. BUILD-TREE: This function takes the NRGSMP as input and creates the TREE by calling the functions below.

c. ADD-NEST: This function takes a node n, then for each state s belonging to the sub-ESMP forming this node, it finds the nests of s $\{M_1,...,M_m\}$ then it creates a new node for each sub-ESMP in the nests, and adds it to the tree with a transition from state s to the appropriate states in the sub-ESMP.

d. MERGE-REG: This function takes the partial TREE as input and merges all nodes that represent the same reg-ESMP.

Before presenting the minor functions, we will introduce the data types.

Data Types

- $E = \{e_0, ..., e_h\}$ is a set of events labels in the given NRGSMP.
- EL is a list of elements of type E.
- A *State* is a class with the following fields: $\left\{\begin{array}{l} K : EL, \\ A : EL, \\ NEST[] : ListofStates \\ ESMP : ListofStates \\ \rightarrow : \mapsto \end{array}\right\}$, where each entry in $NEST[]$ points to a set of states of an ESMP that have this state as one of its in-borders, the field $ESMP$ is a list of states representing the ESMP to which this state belongs, and $\mapsto$ is an array of transitions, i.e. an array of elements of type *Trans* described below.
- A transition *Trans* is a class with the following fields: $\left\{\begin{array}{l} e : E, \\ s' : State \\ s : State \end{array}\right\}$.
- A *Node* is a class with the following fields: $\left\{\begin{array}{l} M : ListofState \\ \rightarrow : ListofTrans \\ \rightarrow^t [S] : \mapsto^t, \\ parent : Node, \\ eqNode : Node, \end{array}\right\}$, where M represents all the states of the sub-ESMP that forms the node, and $\rightarrow$ is a list of all the transitions between the states of the sub-ESMP, $\rightarrow^t [S]$ represents an array of transitions out of each state of the node to the states of the other nodes, so $\rightarrow^t [s]$ is the array whose entries are of type Arc described below, $eqNode$ represents the node to which this node is equivalent (if any), *parent* represents a pointer to the parent of the current node.

- An Arc is a class with the following fields $\begin{Bmatrix} e:E, \\ s':State \\ s:State \\ n:Node \\ n':Node \end{Bmatrix}$, it represents a transitions between two states each in a different node.
- A *ListofState* is a list of elements of type *State*
- A *ListofNode* is a list of elements of type *Node*

Minor Functions:

a. Function CREATE-NODE($S:ListofState, s:State, n:Node$) : void, creates a new node for the tree, the new node represents the sub-ESMP whose states are the set of states from S that are reachable from s, and its parent in the subtree is node n.

b. Function CREATE-ARC(t, r, r', n, n') void, creates a new arc between states r and r' of nodes n and n', respectively, the new Arc represents transition t.

c. Function BELONGS-TREE($r:Node$) : boolean, takes a node r representing a reg-ESMP as input. Its purpose is to check whether the ESMP represented by node r is represented in the TREE by another node say r', then we set $r.eqNode = r'$ meaning that r and r' are equivalent and should be merged, using the function MERGE below.

d. Function CREATE-NEST ($S:ListofState, s:State, n:Node$): Node, takes a set of states that represent the states of an ESMP and one of its in-borders s belonging to node n, and creates a node out of all states in the ESMP that are reachable from s and attaches the nest to the tree by creating all the possible arcs between s and the states of the newly created node.

e. Function IS-REG-ESMP($n : Node$) boolean, takes a node as input and checks whether the node represents a reg-ESMP.

f. Function MERGE ($r, n : Node$) : void, takes two nodes that represent the same reg-ESMP and merges them together making them one node: all arcs leading to the state of r are redirected to lead the state of n.

g. Function ADD (Ψ, N) : void takes a set Ψ and an element N and adds it to the set.

h. Function EXTRACT (Ψ, N) : void, takes a set Ψ and an element N and extracts it from the set .

The minor functions will not be developed as they are straightforward.

Global Variables

- s_0 of type *State* represents the given NRGSMP, every state in this structure (including the starting state) is connected through transitions to a set of other states representing the states that are immediately accessible from this state.
- n_0 of type *Node* represents the HMRP built so far.
- Ω represents the nodes in the tree that need to be further expanded.
- Σ contains all the nodes in the tree that will not be expanded anymore.

4.4.1.3. Definition of the major functions

***Function GET-ESMP( )**: void*

This function will be presented through its main steps. This function searches for all the ESMPs in the NRGSMP s_0, and assigns the correct value to the fields $s.ESMP$ and $s.Nest$ for every state s of NRGSMP s_0, this is done through the following steps:

1. *Find all non-regenerative states in NRGSMP s_0, store them in a set S.*

2. *Divide the states in S into subsets $\{S_1, S_2, ..., S_n\}$ such that for all $i \in \{1,...,n\}$ and for all $s, s' \in S_i$, $A(s) - K(s) = A(s') - K(s')$. And $\{S_1, S_2, ..., S_n\}$ are the maximal subsets with this property.*

3. *Divide each subset of states S_i in the set $\{S_1, S_2, ..., S_n\}$ into further subsets $\Lambda_i = \{S_i^1, S_i^2, ..., S_i^{n_i}\}$ such that, for all $S_i^j \in \{S_i^1, S_i^2, ..., S_i^{n_i}\}$, if $\Delta_i^j = \bigcup_{s' \in S_i^j} K(s')$ then for all $s \in S_i^j$, and for all $e \in \Delta_i^j$, either $e \in K(s)$ or e is inactive in s. And $\{S_i^1, S_i^2, ..., S_i^{n_i}\}$ are the maximal subsets with this property. Then each set of states S_i^j represents the states of an ESMP.*

4. *For each state $s \in S_i^j$ where $i \in \{1,...,n\}$ and $j \in \{1,...,n_i\}$, we set $s.ESMP = S_i^j$,*

5. *For all $i \in \{1,...,n\}$ and all $j \in \{1,...,n_i\}$, we get all the in-borders for the ESMP S_i^j, say $\{s_1,...,s_h\}$, and for each $j \in \{1,...,h\}$, we add S_i^j to the array $s_j.NEST$.*

Function BUILD-TREE (s_0 : State): Node

\\ this function takes the NRGSMP with initial state s_0 as input and returns an HMRP n_0 of type Node as output.

GET-ESMP();

n_0 =CREATE-NODE $(s_0.ESMP, \phi, \phi)$);

$\Omega = \{n_0\}$; $\Sigma = \phi$;

```
While $\Omega \neq \phi$
        EXTRACT $(\Omega, n)$;
        ADD-NESTs$(n, \Omega, \Sigma)$;
MERGE-REG ($\Sigma$);


Function ADD-NESTs ($n : Node, \Omega, \Sigma : ListofNode$): void
$N$ :State;
Repeat for every state $s$ in $n.M$ .
    i=0; integer;
    repeat
        $N =$ CREATE-NEST ( $s.NEST[i], s, n$ );
        If (IS-REG-ESMP($N$) ) and If (BELONGS-TREE($N$) )
            ADD $(\Sigma, N)$
        Else
            ADD $(\Omega, N)$ ;
        $i = i + 1$;
    Until $NEST[i] = null$


Function MERGE-REG ($n_0$ :Node):void
```

```
While $\Sigma \neq \phi$
        EXTRACT($r,\Sigma$)
        MERGE $(r, eqNode(r))$;
                \\ This function takes two nodes of the tree: $r$ and $r'$ that represent equivalent nodes
                and combines them into one node by redirecting the transition coming to $r'$ to lead
                directly to $r$.
```

Theorem 4.9. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an NRGSMP and let $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be the GSMP obtained from G by applying the above algorithm. Then

1. G' is an HMRP and
2. G and G' are structurally bisimilar.

Proof.

To prove that the resulting process is an HMRP, consider the partial tree built by the algorithm. Every node in the tree has exactly one path leading to it. And on this path, each node is an ESMP. Therefore the set of all traces leading to a certain state within a certain node is such that each trace has the form $T_1T_2...T_n$ where each T_i is a trace within a sub-ESMP on the path along the tree.

To prove Point 2, let R be the following relation:

For $s \in S$, $R(s)$ is the set of all states from S' that are created in the algorithm as representing state s. Then for all $r \in R(s)$, s and r have the same active events, and these events lead to states that are related through R. Moreover, $K'(r) = K(s)$ and $F'_r(e) = F_s(e)$ for all $e \in K(s)$.

∎

Figure 9 shows the HMRP obtained by applying the algorithm to the NRGSMP shown in Figure 3. The second digit in the state names in Figure 9 represent the corresponding state in the original NRGSMP.

4.4.2. HMRP to SMP

4.4.2.1. Overview

Given an HMRP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$, we would like to transform G into an SMP, or in other words, we would like to transform every non-regenerative state in G to become a regenerative one. We use the following procedure:

1. For every non-regenerative state s in the HMRP such that T_s^{AvR} is a path $T_s^{AvR} = r_0 \to r_1 \to \ldots \to r_n = s$, we transform s to become regenerative as follows: we first transform r_0 to become regenerative, then r_1 and so on (Theorem 4.6).

2. For every non-regenerative state s in the HMRP such that T_s^{AvR} is a nest $T_s^{AvR} = (N_i(r_r)^*)_{r_r}^{s}$, we transform every state in the nest to become regenerative using the theory of semi-Markov processes (Theorem 4.7)

3. We repeat Steps 1 and 2 until all states are regenerative.

To understand why the above steps actually transform every state in the HMRP to become regenerative, let us consider a non-regenerative state s of the original HMRP G. s will become regenerative by assigning: $F_s(e)(x) = Av\mathrm{Re}\,s(e,s)(x)$ for all $e \in A(s) - K(s)$. From Theorem 4.3 and 4.4, we know that if T_s is a regenerative path with only one regenerative state leading to state s then $Av\mathrm{Re}\,s(e,s) = Av\mathrm{Re}\,s(e,s,T_s^{AvR})$ and T_s^{AvR} is of the form: $T_1(N_1(r_1^{n_1})^*)_{r_1^{n1}}^{r_2^1} T_2(N_2(r_2^{n_2})^*)_{r_2^{n2}}^{r_3^1} ... T_m(N_m(r_m^{n_m})^*)_{r_m^{nm}}^{s}$ where, $T_i = r_i^1 \rightarrow r_i^2 \rightarrow ... \rightarrow r_i^{n_i}$ are paths, $N_i(r_i^{n_i})$ are nests and $(N_i(r_i^{n_i})^*)_{r_i^{ni}}^{r_{i+1}^1}$ are set of traces belonging to N_i. Let us apply the steps above to the HMRP and focus on its impact on state s:

- Step 1 transforms every state in the path T_1 to become regenerative, starting from state r_1^1, then state r_1^2, until state $r_1^{n_1}$ (note that r_1^1 is already regenerative).

- Step 2 transforms all states in N_1 to become regenerative. Note that the in-border $r_1^{n_1-1}$ was already transformed to be regenerative in the previous step.

- Next we repeat step 2 on path T_2. (Note that r_2^1 was already transformed to become regenerative in the previous step).

- And so on … until s becomes regenerative.

4.4.2.2. Algorithm 2

Traverse the HMRP G and transform every state s to become regenerative (starting from the starting state) as follows:

- If s is not part of any nest, i.e. if every trace to s has a postfix $t : s' \xrightarrow{e} s$, we transform s to become regenerative using Theorem 4.6 (note that s' should have been transformed into a regenerative state in previous steps).

- If s is part of a nest $(N_i(r_i)^*)_{r_i}^{s}$, we transform every state in the nest to become regenerative using Theorem 4.7 (note that r_i should have been transformed into a regenerative state in previous steps).

The SMP obtained would have the same states as the HMRP, and it s-simulate the HMRP, the relation that establishes the s-simulation is the identity relation.

Theorem 4.10. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a NRGSMP and let $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be the HMRP obtained from G by applying Algorithm 1. And let $G'' = (S', s'_0, \mathrm{E}', F'', \mapsto', K'')$ be an SMP obtained from G' by applying Algorithm 2. Then G'' s-simulates G.

Proof. The proof is easily deduced from the way we constructed the SMP.

■

4.4.3. Application

As an application, consider State 1-1 in Figure 9. We will attempt to find $Av\,\mathrm{Re}\,s(a,1-1)$. Note that T_{1-1}= 0-0 → 1-1 is the defining trace for state 1-1, the ESMP to which state 1-1 belongs is $1-1(\xrightarrow{r} 2-2 \xrightarrow{s} 1-1)^n \xrightarrow{r} 2-2 \xrightarrow{f} 3-3)$, and

$T_{1-1}{}^{AvR} = \{0-0 \xrightarrow{d} 1-1(\xrightarrow{r} 2-2 \xrightarrow{s} 1-1)^n \mid n$ any integer$\}$. So from Theorem 4.2, $Av\,\mathrm{Re}\,s(a,1-1) =$ $Av\,\mathrm{Re}\,s(a,1-1,T_{1-1}{}^{AvR})$. Note that $T_{1-1}{}^{AvR}$ is composed of one nest $N(0-0)$ (shown in Figure 12 (a)). So to find $Av\,\mathrm{Re}\,s(a,1-1,T_{1-1}{}^{AvR})$ we can apply Theorem 4.7. Note that the SMP $\vec{N}(0-0)$ is shown in Figure 12 (b), (recall that $\vec{N}(0-0)$ is formed by adding the missing transitions governed by the events from $K(1-1) \cup K(2-2)$).

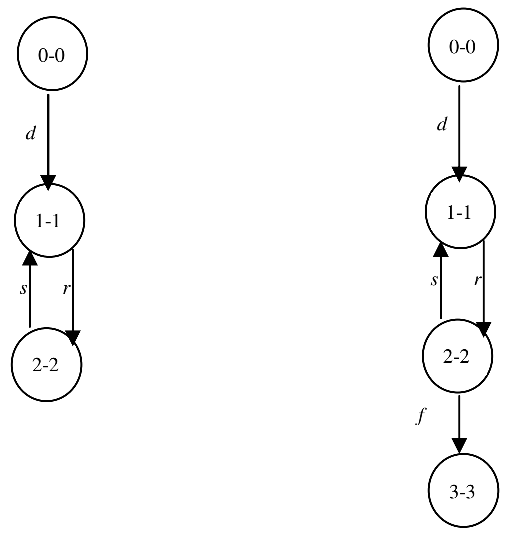
We have that: $Av\,\mathrm{Re}\,s(a,1-1,T_{1-1}{}^{AvR}) = Ave\,\mathrm{Re}\,s(a,1-1,N(0-0)*^{1-1}_{0-0})$

And with a straightforward application of Theorem 4.7 we get:

$Ave\,\mathrm{Re}\,s(a,1-1,N(0-0)*^{1-1}_{0-0}) =$

$$\int_0^\infty [\frac{(1-F_{0-0}(a)(x'))[1-\frac{L_{3-3}^{\vec{N}(0-0)}(x')}{x'}]\frac{d\pi_{1-1}{}^{N(0-0)}(x')}{dx'}}{\int_0^\infty (1-F_{0-0}(a)(x''))[1-\frac{L_{3-3}^{\vec{N}(0-0)}(x'')}{x''}]\frac{\pi_{1-1}{}^{N(0-0)}(x'')}{dx''}dx''} \; \frac{F_{0-0}(a)(x'+x)-F_{0-0}(a)(x')}{1-F_{0-0}(a)(x')}]\,dx'$$

The final step would be to set $F_{1-1}(a)(x) = Av\mathrm{Re}s(a,1-1,T_{1-1}{}^{AvR})(x)$.



a. SMP $N(0-0)$ b. SMP $\vec{N}(0-0)$

Figure 12. SMPs $N(0-0)$ and $\vec{N}(0-0)$

4.5. Practical Limitations

4.5.1. Space Complexity

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an NRGSMP. Assume that all ESMPs in G are strongly connected, and let $\{M_1, M_2, ..., M_q\}$ be the set of strongly connected ESMPs in G. (Note that if the ESMPs are not strongly connected then we take the strongly connected ESMP parts; refer to Theorem 4.1, and the analysis would only be slightly different). Let

$G'' = (S', s'_0, \mathrm{E}', F', A', \mapsto'', K')$ be the HMRP obtained by applying Algorithm 1. Let $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be the HMRP obtained from G by applying Algorithm 1, but instead of executing the function MERGE-REG (Σ) at the end, we delete all regenerative states in the set Σ (refer to Section 4.4). Then G' has the same number of states as G'', but G' has no regenerative cycles. Let R be the s-simulation from G to G''. We would like to determine $|S'|$. For that, we will consider G' rather than G'' as the absence of regenerative cycles in G' renders it easier to work with.

We need the following notation:

1. t is the maximum number of transitions out of a state in the HMRP (or NRGSMP), that do not lead to a regenerative state, we call it the branching factor
2. m is the maximum number of states among the ESMPs of G, if G has no ESMPs then $m = 1$.
3. S^R is the set of regenerative states in G
4. S^{SMP} is the set of states in G such that $|s.ESMP| > 1$
5. S^{rem} are the remaining states in G: $S^{rem} = S - (S^{SMP} \cup S^R)$
6. $|S| = n$ is the number of states in G

Lemma 4.3. We define a relation R' on the states of G' as follows:

$sR's'$ if $s, s' \in R(M_i)$ for some $i \in \{1, ..., h\}$. Then R' partitions the set S' into distinct subsets S'/R'. We divide the set S'/R' into two distinct subsets S'_1/R' and S'_2/R' where S'_1/R' contains all sets containing exactly one state (i.e. all sets from S'/R' that consist of only one element). Let $S'_1 = R^{-1}(S'_1/R')$. Then

- $|S'_1| = |S'_1/R'|$

- $|S'_2/R'|$ is equal to the number of ESMPs in the HMRP G', in other words, every state in $|S'_2/R'|$ represents an ESMP from the set $\{M_1, M_2, ..., M_h\}$
- $|S'| < |S'_1/R'| + m|S'_2/R'| =. |S'_1| + m|S'_2/R'|$.

Proof. Straightforward

∎

Lemma 4.4. Let $H = (S'/R, \mapsto'/R)$ be a labeled transition system, where S'/R is defined as above and for all $t : s \to s' \in \mapsto$, $t : s/R \to s'/R \in \mapsto'$, then H is a tree.

Proof. Note that because of our assumption about ESMPs, the tree *H* is the tree of nodes that we built in Algorithm 1.

∎

Now, we divide the set $\{M_1, M_2, ..., M_q\}$ into two subsets, say $\{M_1, M_2, ..., M_h\}$ and $\{M_{h+1}, ..., M_q\}$, such that all the in-borders of the ESMPs in $\{M_1, M_2, ..., M_h\}$ are regenerative.

We would like to determine the number of states in S'/R, and for that we assume that the branching factor for H is t', the number of leaf nodes to be l (i.e. the number of different branches in the tree),

Lemma 4.5. The tree defined in Lemma 4.4 has the following properties:

- $l < (mt)^{|S^{rem}| + (q-h)}$
- $S'_1 < (|S^R| + l)|S^{rem}|$
- $S'_2/R = l(q-h) + |S^R|h$

Proof. The proof could be deduced from the following observations:

- Every state in S^R is represented only once in the set S'_1.
- Let s be a state in S^{rem}, then s is represented at most once in a branch of H (otherwise if it is represented twice, then there would be an execution of a non-regenerative cycle from G, but s is not part of any ESMP, a contradiction)
- Let M_i be an ESMP in $\{M_{h+1},...,M_q\}$, then M_i is represented by at most one state in every branch of H and once per regenerative state i.e. by a total of $|S^R|+l$ times. (recall that M_i is strongly connected).
- Let M_i be an ESMP in $\{M_1, M_2,..., M_h\}$, then M_i is represented at most once per regenerative state (i.e. by a total of $|S^R|$ times.).

∎

From the above theorem, we conclude that $|S'/R| < (|S^R| + (mt)^{|S^{rem}|+(q-h)})|S^{rem}|$ and $S'_2/R = (mt)^{|S^{rem}|+(q-h)}(q-h) + |S^R|h$ and

$$n_T = (|S^R| + (mt)^{|S^{rem}|+(q-h)})|S^{rem}| + m[(mt)^{|S^{rem}|+(q-h)}(q-h) + |S^R|h]$$

$$n_T = O[(mt)^{|S^{rem}|+(q-h)})(|S^{rem}| + m(q-h))]$$

Theorem 4.11. If $G' = (S', s'_0, E', F', A', \mapsto', K')$ implements the enabling restriction then

- $S^{rem} = \phi$ and
- $q = h$

Idea behind the Proof. Let $s \in S'$ be a non-regenerative state, then $A(s)$ would consist of one non-exponentially distributed event, say g, and several exponentially distributed ones, say $e_1,...,e_v$, so state s has $v+1$ transitions out of it corresponding to events $e_1,...,e_v$ and g, to states $s_1,...,s_v,s_{v+1}$ respectively. And the sub-GSMP formed by s and $s_1,...,s_v$ and the transitions $e_1,...,e_v$ is an SMP. Note that if $e_1,...,e_v$ is empty, then for any state r such that s is accessible from r by one transition, $s \in ESMP$ of r. Point 2 is straightforward.

∎

Note however that if $S^{rem} = \phi$ and $q = h$, then the NRGSMP need not implement the enabling restriction, an example is depicted in Figure 10.

Now if $|S^{rem}| = 0$, i.e. if all states are either regenerative or belong to a non-trivial ESMP, then $n_T = O[(mt)^{(q-h)})(m(q-h))]$, so if $q-h$ is small, i.e. if the number of ESMP's with at least one non-regenerative in-border is small (say 2), then this method would have an acceptable space limitation. Now, if $|S^{rem}| = 0$ and $q = h$, then $n_T = O(mh|S^R|) = O(|S - S^R||S^R|)$.

For the example in Figure 10, $mt = 1$ and $h = q = 0$, hence $n_T = |S^R| + |S^{rem}| = n$

4.5.2. Time Complexity

We define the basic operations as:

- equality checking,
- assignment statements,
- unions and subtractions of two sets, and
- additions and subtractions of two integers.

The complexity of the basic operations is assumed to be a fixed constant (we use the value 1). Moreover, we assume that the integral calculation involves the evaluation of an expression a large number of times. We call this number c. The evaluation of the expression has its own complexity which must be taken into account.

The time complexity will be presented for each algorithm in the following:

<u>Algorithm1</u>: In this algorithm, the total number of nodes that we create is n_T. We first execute function GET-ESMP. Then for every node that we create, we execute function ADD-NEST. Then after creating all the nodes, we call function MERGE-REG. So if we denote by A, B and C the complexities of ADD-NEST, MERGE-REG and GET-ESMPs respectively, the total complexity of Algorithm 1 would be:

$$A_1 = A + B + C \qquad (1)$$

where

- A: ADD-NEST calls CREATE-NEST, whose complexity is m, a maximum of mt times, and function BELONGS-TREE , whose complexity is n_T, a maximum of t times, so its total complexity is $A = (tn_T{}^2 + tmn_T)$
- B:MERGE-REG executes MERGE, which has a complexity of t, $|\Sigma|$ times. But $|\Sigma|$ is less than the total leaf nodes in the tree, i.e. $|\Sigma| < (mt)^{|S^{rem}|+(q-h)}$ and $B = t|\Sigma| < t(mt)^{|S^{rem}|+(q-h)}$.
- C: GET-ESMP, this function has a complexity of $C = 4n_T$ as each of the steps 1, 2, 3 and 5 has a complexity of n_T

Note that many of the functions are not referred to here as they have a complexity of 1.

Equation (1) then becomes:

$$A_1 = (tn_T{}^2 + tmn_T) + t(mt)^{|S^{rem}|+(q-h)} + 4n_T$$

$$= O(tn_T{}^2)$$

Algorithm2: This algorithm involves the following computations:

1. For every $N_i(s)$, determine $\vec{N}_i(s)$: note that we have a maximum of t transitions out of a state, so if n_i is the number of states in the nest $N_i(s)$ then the complexity for determining $\vec{N}_i(s)$ is: $O(\sum_{i=1}^{j} n_i t)$.

2. For every $N_i(s_i)$ determine the steady state probability for $N_i(s_i)$ and the expected time spent in an absorbing state of $\vec{N}_i(s_i)$ for a given interval. If we have *j* nests $N_i(s_i)$ such that $N_i(s_i)$ has n_i states and $\vec{N}_i(s_i)$ have say n'_i states, for all $i \in \{1,..., j\}$ then the complexity for the this step is less than $O(\sum_{1}^{j} {n'_i}^3 + 2c\sum_{1}^{j} {n'_i}^3) = O(2cm^3)$.

3. For every state, applying Theorem 4.6 or 4.7: the complexity of this step is $O(2cn_T)$

So the total complexity for Algorithm 2 is:

$$O(+\sum_{i=1}^{j}(n_i t) + (2cm^3) + (2cn_T) = O(2cm^3 + 2cn_T)$$

To summarize, the total time complexity for Algorithms 1 and 2 would be $= O(\ t{n_T}^2 + (2cm^3 + 2cn_T))$

Now, If $|S^{rem}| = 0$, then the time complexity becomes $O(\ t{n_T}^2 + (2cm^3 + 2cn_T))$ where $n_T = O[(mt)^{(q-h)})(m(q-h))]$.

And if $S^{rem} = \phi$ and $q = h$, then the time complexity becomes $O(t\,|S||S - S^R|^2 + 2cm^3 + 2c\,|S||S - S^R|)$

4.6. Conclusion

In the literature, there are two main methods that attempt to analyze GSMPs: the regenerative and the supplementary variable methods. Both methods can be applied to a subset of GSMPs, those that implement the "enabling restriction" meaning that only one non-exponentially distributed clock can be active at any given time. GSMPs with the enabling restriction can only have regenerative or Markovian cycles. To calculate the steady state probabilities of the different states of a GSMP, with the above restriction, at time t, the regenerative method has a worst case of $O(|S|^2)$ space complexity and $O(|S|^4)$ time complexity [40]. The method of supplementary variables has a worst case of $\approx O(q^g |S|^2)$ time complexity and $O(|S^E| + c\sum_{g \in T^G} |S^g| + \sum_{g \in T^G} |S^g|^2)$ space complexity where S^E is the set of states in which only exponential transitions are enabled, S^g is the set of states in which the non-exponential transition g is enabled, c denotes the time for integral calculation, and q^g the absolute maximum diagonal entry for Q^g (refer to chapter 3) [40].

In this chapter, we presented an algorithm for finding an analytical solution for a subset of GSMPs. In this subset we allow non-exponentially distributed events to be initialized anytime, but we impose a restriction on the type of cycles in the GSMPs: they all have to be near semi-Markovian or regenerative cycles. The time and space complexity of the algorithm presented in this chapter is exponential in the number of states in the set S^{rem}, i.e. states that are neither regenerative nor belong to a non-trivial ESMP, and in the number of maximal strongly connected ESMPs $(q-h)$ that have at least one non-regenerative in-border. But, when applied to GSMPs whose states either belong to an ESMP with regenerative in-border or are regenerative, the algorithm becomes $O(|S|^R |S - S^R|)$ in space complexity, where S^R is the set of regenerative states, and $O(2cm^3 + t|S|^2 + 2c|S|^2)$ in time complexity, where m is the maximum number of states of the ESMPs and t is the branching factor of the NRGSMP.

The exponential factor in the complexity of the algorithm limits its real applicability to GSMPs with a small number of states in the set S^{rem}, and a small value of $q-h$, this set of processes contains GSMPs that were not covered by previous methods.

Chapter 5: Time Preserving Simplification for GSMPs

5.1. Introduction

In this chapter, we explore a method to reduce the time and space complexities of the algorithm presented in the previous chapter. Given an GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$, we will delete states in the GSMP G while preserving the passage of time distribution between pairs of non-deleted states.

So given a GSMP G, and assume we deleted n states out of G, say $\{s_1, s_2, ..., s_n\}$ and let $G' = (S', s_0, \mathrm{E}', F', A', \mapsto', K')$ be the resulting GSMP, where $S' = S - \{s_1, s_2, ..., s_n\}$. Now let s, s' be any two states in S', then the passage of time distribution between s and s' in G' would be equal to the passage of time distribution between s and s' in G. However, the sojourn time in the states of the set S' (and hence the TSP) may not be preserved , in fact, all the states that have an outgoing transition to a state in the set $\{s_1, s_2, ..., s_n\}$, will have a different TSP in G and G', while for all other states the TSP is preserved. So if we are interested in the TSP of a subset S'' of states in the GSMP G, then we can use the simplification presented in this chapter to delete all states that satisfy the above two conditions and that are not directly accessible from the set S''.

The *passage of time equivalence* is not new, it was introduced by Bradley in [14] in the context of SMPs. So we use the same definition and extend it in the context of GSMPs. The definitions of equivalences are presented in the next section, followed by the simplification steps, then the algorithm is presented and finally the complexity of the algorithm is discussed with some concluding remarks.

5.2. Definition of Equivalences

Definition 5.1: Passage-time equivalence

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be two GSMPs. Let $\Sigma \subseteq S$ and $\Sigma' \subseteq S'$ be of the same cardinality. We say that G and G' are *passage-time equivalent* over Σ and Σ', written $G \overset{\Sigma,\Sigma'}{\sim} G'$, if there exists a one-to-one correspondence between Σ and Σ': $f : \Sigma \to \Sigma'$ such that $s_0 \in \Sigma$, $s'_0 \in \Sigma'$, and $f(s_0) = s'_0$, and if $s, s' \in \Sigma$ then the passage-time distribution from s to s' in G is identical to the passage-time distribution from $f(s)$ to $f(s')$ in G'.

Before presenting the next definition we introduce the notation $P(s(x) \mid s_0(0))$ that stands for the probability of being in state s at time x given that we were in state s_0 at time 0 (in other words, it is the TSP for state s at time x)

Definition 5.2: Transient state equivalence

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be two GSMPs. Let $\Sigma \subseteq S$ and $\Sigma' \subseteq S'$ be of the same cardinality. We say that G and G' are *transient state equivalent* over Σ and Σ', written $G \overset{\Sigma,\Sigma'}{\approx} G'$, if there exists a one-to-one correspondence between Σ and Σ' $f : \Sigma \to \Sigma'$ such that if $s \in \Sigma$ then $P(s(x) \mid s_0(0)) = P(f(s)(x) \mid s'_0(0))$.

5.3. Simplification Technique

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, we can delete any state s from the GSMP provided s has the following properties:

1. s belongs to an ESMP M, and
2. for every in-border s'_0 of M, s'_0 is not directly connected to s, i.e. there exists no transition between s'_0 and s, and
3. $K(s)$ is a singleton: $K(s) = \{f_2\}$, i.e. there exists only one outgoing transitions from state s relative to the ESMP M, and for all $s' \in M$, if $s' \xrightarrow{e} s$, then $K(s') = \{e\}$, and
4. for all $r \in S$, if r is directly connected to s and if $s \xrightarrow{e_i} s_i$ for some $s_i \in S$ and $e_i \notin K(s)$, then $r \xrightarrow{e_i} s_i$

Note that after deleting a state s, following the algorithm presented later in the chapter, the distribution of time to travel between non-deleted states, as well as the transient state probabilities for all states that are not directly connected to s would be preserved, while the TSP for the states that are directly connected to s increases as these states share among them the TSP of state s (refer to Theorem 6.2). We consider the following two scenarios:

1. Assume we are interested in a performance study that involves the SSP of a set $\Sigma \subseteq S$ of states in G, for example, if we are interested in the probability of

failure, then Σ should contain all the states that represent *failure in steady state*. Then we can delete all states in the set $S-\Sigma$ that satisfy the above four conditions and that are not directly accessible from states in the set Σ (i.e. there exists no transition from a state in Σ to any deleted state). The aim of state deletion is the facilitation of the performance analysis by having a smaller state space. The simplified GSMP $G'=(S',s'_0,\mathrm{E}',F',A',\mapsto',K')$ would then be transient state equivalent to G, $G\overset{\Sigma,\Sigma}{\approx}G'$, and as a result, the set of states Σ would preserve their TSP and hence their SSP.

2. Alternatively, if we are interested in a performance study that involves the distribution of time to travel between the states of a set $\Sigma\subseteq S$, for example, if we are interested in the distribution of time until a failure occurs then Σ would contain the starting and the fail states, then we delete all states in the set $S-\Sigma$ that satisfy the four conditions above. The simplified GSMP $G'=(S',s'_0,\mathrm{E}',F',A',\mapsto',K')$ would then be passage-time equivalent to G, $G\overset{\Sigma,\Sigma}{\sim}G'$.

To illustrate, for any ESMP M of G with state space $S_E\subseteq S$, such that $S_E\cap(S-\Sigma)\neq\phi$, let $\{e_1,...,e_n\}=A(s)-K(s)$ for all $s\in S_E$. If there exists a state that $s\in S_E\cap(S-\Sigma)$ that satisfies the four conditions above, then we apply the sequential reduction described below to s (note that if we are interested in a scenario similar to scenario 1 above, then s should not be directly accessible from any state in Σ):

Sequential Reduction (t_1,t_2): Given two sequential transitions to and from state s: $t_1=r\xrightarrow{f_1}s$ and $t_2=s\xrightarrow{f_2}r'$ (refer to Figure 13), we would like to aggregate t_1 and t_2 so as to form a single transition. Note that $K(s)=\{f_2\}$. Assume for simplicity of presentation that $A(s)-K(s)=\{e_1\}$. To delete state s in Figure 13, we create a new event f_1f_2, which is the concatenation of the two events governing transitions t_1 and t_2, then we set $K(r)=\{f_1 f_2\}$, and $A'(r)=A(r)\cup(f_1 f_2\}-\{f_1\}$ and assign to event $f_1 f_2$ the following distribution:

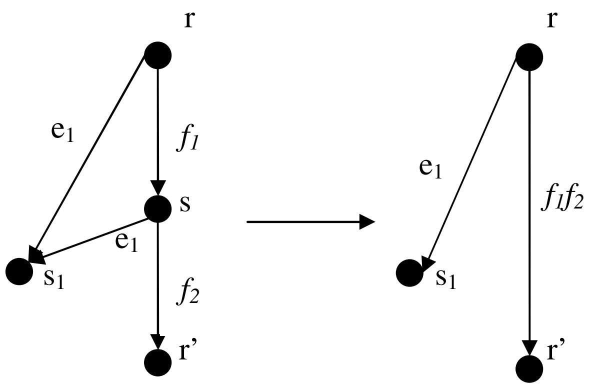


Figure 13. Sequential reduction

$F'_r(f_1 f_2)(x)=\int_0^\infty \frac{dF_r(f_1)(x')}{dx'}F_r(f_2)(x-x')dx'$. We note that $F'_r(f_1 f_2)(x)$ is the convolution of the distributions associated with events f_1 and f_2, and for that reason, the distribution of time needed to travel from state r to state r' would be preserved.

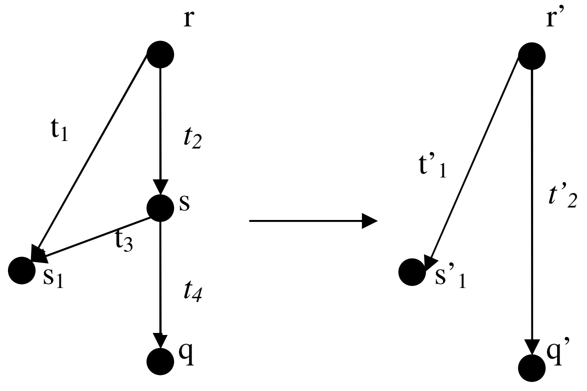
Note that, after the deletion of state s, the sojourn time for state r changes. In fact, given we are in state r, if trace $f_1\, f_2$ occurs in G and G', then the elapsed time since entering state r until trace $f_1\, f_2$ occurs is divided among states r and s in G, while in G' the total time is spent in state r. In other words, part of the TSP of state s is taken by state r, that part is: $P^G(s(x)\mid s_0(0)\wedge t_1)$, i.e. the probability of being in state s at time t given that we reached state s by following transition t_1 is now part of the TSP of state r.

5.3.1. Illustration

To illustrate the above transformation consider the two semi-Markov processes shown below, where:

- t_1, t'_1 and t_3 are exponentially distributed transitions with rate equal to λ.
- t_2 and t_4 are both exponentially distributed with rate μ.
- t'_2 is the following Erlang distribution: $F(x) = 1 - e^{-\mu x} - \mu x e^{-\mu x}$ (in other words, it is the convolution of 2 exponential distributions with rate μ).

We say *that a transition takes x time units to occur* if the clock associated with the transition lives for a total of *x* time units (since it is initialized until it expires). In the Figures below we have two SMPs, so clocks that do not expire when we move to a new state are disabled.



We will prove that $\frac{dP(s_1(x) \mid r(0))}{dx} = \frac{dP(s'_1(x) \mid r'(0))}{dx}$.

Note that, for the SMP on the left, entering state s_1 at time x when the process was in state *r* at time 0, means that

1. either transition t_1 took exactly x time units to occur, and transition t_2 did not occur in the interval $[0, x]$. Or
2. transitions t_2 occurred out of state *r* followed by transition t_3 and both transitions combined took exactly x time units (so if transition t_2 took exactly x' time units to

occur then that means that transition t_1 did not occur in $[0,x']$. It also means that transition t_3 took exactly $x-x'$ time units to occur and that transition t_4 did not occur in $[0,x-x']$.

For the SMP on the right, entering state s'_1 at time x means that transition t'_1 took exactly x time units to occur, and transition t'_2 did not occur in the interval $[0, x]$.

Hence,

- $\frac{dP(s_1(x) \mid r(0))}{dx}$ = (probability that transition t_1 takes exactly x time units to occur while transition t_2 takes more that x time units)/dx +(probability that transition t_2 takes exactly x' time units to occur while transition t_1 takes more that x' time units, and transition t_3 takes $x-x'$ time units to occur while transition t_4 takes more than $x-x'$, for some x' in [0, x])/dx

  $$= (\lambda e^{-\lambda x} e^{-\mu x}) + \left(\int_0^x \mu e^{-\mu x'} e^{-\lambda x'} \lambda e^{-\lambda(x-x')} e^{-\mu(x-x')} dx'\right)$$
  $$= \lambda e^{-(\lambda+\mu)x} + \lambda\mu \int_0^x e^{-(\lambda+\mu)x} dx'$$
  $$= \lambda e^{-(\lambda+\mu)x} + \mu\lambda x e^{-(\lambda+\mu)x}$$

- $\frac{dP(s'_1(x) \mid r'(0))}{dx}$ = (probability that transition t'_1 takes exactly x time units to occur while transition t'_2 takes more that x time units)/dx.

  $$= \lambda e^{-\lambda x}(e^{-\mu x} + \mu x e^{-\mu x})$$
  $$= \lambda e^{-(\lambda+\mu)x} + \mu\lambda x e^{-(\lambda+\mu)x}$$

Hence these probabilities are preserved.

5.3.2. Overall Algorithm

The algorithm takes as input a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and a state s satisfying the three conditions of the previous section. It then outputs a GSMP $G' = (S - \{s\}, s_0, \mathrm{E}', F', A', \mapsto', K')$.

We assume we have the same data structure as in the previous chapter.

```
Function main():void
Repeat for every path $t_1 t_2$ having s as the middle state.
    SEQUENTIAL-REDUCTION($t_1, t_2$);
Output the resulting GSMP;
End.
```

Theorem 5.1. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP and let $G' = (S - \{s\}, s_0, \mathrm{E}, F', A, \mapsto', K)$ be the GSMP obtained by applying the algorithm above to a state $s \in S$, let $S' = \{s_1, ..., s_m\}$ be the set of states $\in S$ that are directly connected to s in G then

1. $G \overset{S-\{s\},S-\{s\}}{\sim} G'$, and
2. $G \overset{S-S',S-S'}{\approx} G'$

Proof.

For the proof of (1), consider the particular case of Figure 13, then we need to prove that $P(r'(x) \mid r(0) \wedge r \xrightarrow{f_1} s \xrightarrow{f_2} r')$ and $P(s_1(x) \mid r(0) \wedge r \xrightarrow{e_1} s_1)$ are preserved after the transformation. Here $P(r'(x) \mid r(0) \wedge r \xrightarrow{f_1} s \xrightarrow{f_2} r')$ is the probability of being in state r' at time x, counting from when the system entered state r, and that trace $r \xrightarrow{f_1} s \xrightarrow{f_2} r'$ was followed to move from r to r'.

- In both GSMPs, $P(r'(x) \mid r(0) \wedge r \xrightarrow{f_1} s \xrightarrow{f_2} r')$ is a function of:
    - the ability of f_2 (preceded by f_1) to occur at some time $x' \in [0, x]$ (i.e. a function of the distribution $F'_r(f_1 f_2)$), and
    - on the distribution of the soujourn time in state r' ($\zeta(r')$ has to be $\geq x - x'$), and
    - on the probability that the clock of event e_1 expires after more than x' time units (counting from when state r was entered).

  Note that all these probabilities are the same in both GSMPs

- In the GSMP of the left of Figure 13, $P(s_1(x) \mid r(0) \wedge r \xrightarrow{e_1} s_1)$ is a function of
    - the ability of e_1 to occur either before f_1 does, or after f_1 and before f_2. In other words, e_1 has to occur at any time before the combined event $f_1 f_2$ occurs. I.e. the clock associated with event e_1 needs to expire at some time $x' \in [0, x]$ while that of the combined event, (i.e. the clock whose distribution is $F'_r(f_1 f_2)$) has to expire after x'.

To prove point (2), assume for simplicity that state s is only directly accessible from a state $r \in S$, then for all $v \in S - \{S'\}$,

$$P(v(x) \mid s_0(0))$$

$$= P(r(x') \mid s_0(0) \wedge \text{without visiting } s)\ P(v(x) \mid r(x'))$$

Note that $P(r(x') \mid s_0(0) \wedge \text{without visiting } s)$ is unchanged between G and G'. Moreover, we can deduce from point (1) above that $P(v(x) \mid r(x'))$ is also unchanged.

∎

Theorem 5.2. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP and let $G' = (S - \{s\}, s_0, \mathrm{E}, F', A, \mapsto', K)$ be the GSMP obtained by applying the algorithm above to a state $s \in S$, let $\{s_1, ..., s_m\}$ be the set of states $\in S$ that are directly connected to s in G then

$$P^{G'}(s_i(x) \mid s_0(0)) = P^{G}(s_i(x) \mid s_0(0)) + P^{G}(s(x) \mid s_0(0) \wedge t_i)$$

where $P^{G}(s(x) \mid s_0(0) \wedge t_i)$ is the probability of being in s at time x given that we were in s_0 at time 0, and that we followed a path ending with transition t_i to reach s.

Proof. Assume that $s_i \xrightarrow{f_i} s \xrightarrow{f'_i} s'_i$ is a trace in G, where $f'_i = K(s)$, then this would be transformed into the following trace in G': $s_i \xrightarrow{f_i f'_i} s'_i$. Note that $P(s'_i(x) \mid s_i(0))$ is preserved, and hence the time until we reach state s'_i from state s_i is preserved. Hence the time process G spends in s given that trace $s_i \xrightarrow{f_i} s \xrightarrow{f'_i} s'_i$ will occur is added to the time spent in s_i after the deletion of state s.

∎

5.3.3. Complexity

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be a GSMP, let $M = (S_E, s'_0, \mathrm{E}, F, A, \mapsto, K)$ be an ESMP of G. The time required to delete one state $s \in S_E$ in the GSMP such that $\{e_1, ..., e_n\} = A(s) - K(s)$ is $cn \mid S_E \mid$ (where c represents the complexity for integral calculation) in the worst case, i.e. if every state in the ESMP is connected to state s.

Given an NRGSMP that we want to analyze by transforming it to an SMP, then applying the simplification algorithm to some of its states is very useful in decreasing the complexity of the transformation. In fact, deleting one state from an ESMP in the NRGSMP, leads to an HMRP with as much as $t^{|S^{rem}|+(q-h)} \mid S^{rem} \mid$ fewer states, (refer to Section 4.5). Note however that, to be able to delete a state from the NRGSMP, the state needs to satisfy the four restrictions stated in Section 5.3. And that the performance measures that are preserved are the ones that depend only on the SSPs of the states that are not directly connected to the deleted states.

Chapter 6: Mean Passage-Time Equivalence for SMPs

6.1. Introduction

In this chapter we introduce an equivalence: “mean passage time equivalence” and a state simplification technique for SMPs. The simplification technique allows us to delete states from the SMP while preserving the average of the passage-time distribution between pairs of non-deleted states. As far as we are aware, the only existing simplification in the context of semi-Markov processes was introduced by Bradley in 2002 [14]. In his paper, Bradley introduced a simplification technique that preserves the exact passage-time distributions between pairs of non-deleted states; this in stochastic terms is a very strong equivalence, the two models under comparison should have strong similarity. In this chapter, the equivalence is less restrictive; processes would still be equivalent if they have the same average of passage time distribution between states rather than exact distributions; and the simplification procedure requires less time as its steps are straightforward. This equivalence is useful when the user is only interested in mean passage-time delays and not actual distributions. The *"mean-passage time equivalence"* preserves all performance measures that depend on mean passage time such as reliability and availability. So if we are interested in such measures, the simplification technique would help us reduce the size of the SMP and hence the complexity for the performance evaluation procedure. The simplification technique also preserves the SSP of a subset of the non-deleted states: these are the states that do not directly lead to a deleted state (through one transition).

This chapter is structured as follows: we first define the equivalence, then we present the simplification steps, then the whole technique is justified and presented formally, and finally, we study the effects of the simplification on our original NRGSMP, and discuss some of the performance aspects that are preserved.

6.2. Equivalence Definition

Recall that a semi-Markov process G is a tuple $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$, (note that $A(s) = K(s)$ for all $s \in S$, and that's why A was omitted). Informally, two semi-Markov processes *M* and *N* are mean-passage time equivalent if the passage-time distributions between states of *M* (or a subset of states of *M*) have the same mean as a certain reordering of passage time distributions between states of *N* (or a subset of states of *N*), formally:

Definition 6.1: Mean passage-time equivalence

Let *G*= $(S, s_0, \mathrm{E}, F, \mapsto, K)$ and *G'*= $(S', s'_0, \mathrm{E}', F', \mapsto', K')$ be two SMPs. Let $M \subseteq S$ and $N \subseteq S'$ be of the same cardinality: $|M| = |N|$. Let G^* be the SMP obtained from G by applying the algorithm in Section 6.3.2. repeatedly on the set of states $S - M$. Similarly, Let G'^* be the SMP obtained from G' by applying the algorithm in Section 6.3.2. repeatedly on the set of states $S' - N$. Then we say that *G* and *G'* are *mean passage-time equivalent* over *M* and *N*, written $G \overset{M,N}{\equiv} G'$, if G^* and G'^* are isomorphic up to state relabelling.

Definition 6.2: Steady state equivalence

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and $G' = (S', s'_0, \mathrm{E}', F', A', \mapsto', K')$ be two GSMPs. Let $\Sigma \subseteq S$ and $\Sigma' \subseteq S'$ be of the same cardinality. We say that G and G' are *steady state equivalent* over Σ and Σ', written $G \overset{\Sigma,\Sigma'}{\cong} G'$, if there exists a one-to-one correspondence between Σ and Σ' $f : \Sigma \to \Sigma'$ such that if $s \in \Sigma$ then $\pi(s) = \pi(f(s))$.

For more information, refer to [14].

6.3. Simplification Technique

In this section, we follow the same steps as in [14]. The only difference is that we are interested in preserving the mean and not the actual distributions.

6.3.1. Basic Reduction Steps

Given an SMP $G = (S, s_0, \mathrm{E}, F, \mapsto, K)$ and a set of states M in S, we would like to delete all states in the set (S-M) such that the resulting SMP $G' = (M, s'_0, \mathrm{E}', F', \mapsto', K')$ is mean passage-time equivalent over M, i.e. $G \overset{M,M}{\equiv} G'$.

For every transition $t : s \to s'$ we designate by the tuple (a, p) the mean of the passage-time distribution, a, for transition t and the transition probability, p, respectively. We call the tuple (a, p) the coordinates for transition t. In what follows, we will first present the basic algorithm steps, how and when these steps are used will be detailed later. We denote by $t.e$ the event associated with transition t.

Sequential Reduction (t_1,t_2)**:** Given two sequential transitions t_1 and t_2 (refer to Figure 14), we would like to delete them so as to form a single transition t_3 such that $t_3.e = t_1.e\ t_2.e$; note that the event $t_1.e\ t_2.e$ is the concatenation of the events $t_1.e$ and $t_2.e$. So if (a_1, p_1) and (a_2, p_2) are the coordinates for transitions t_1 and t_2, respectively, then $(a,p)=(a_1 + a_2, p_1 p_2)$ are the coordinates for transition t_3 (note that a is actually the mean of the distribution calculated in Section 5.3, see also [14]).

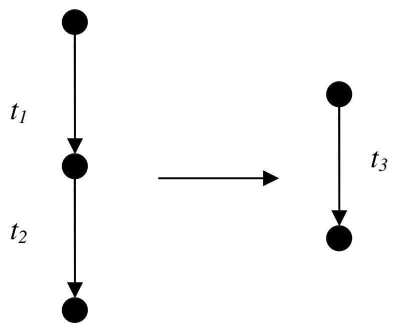


Figure 14. Average sequential reduction

Alternate Reduction (t_1,t_2)**:** The sequential reduction step might create more than one transition, say t_1 and t_2, having the same starting and ending states, and governed by the same event, i.e. $t_1.e = t_2.e$, the scenario is shown in Figure 15. Therefore an alternate reduction step may be necessary. Given two alternate transitions t_1 and t_2 with $t_1.e = t_2.e$, we would like to delete them so as to form a single transition t_3. So if (a_1, p_1) and (a_2, p_2) are the coordinates for transitions t_1 and t_2, respectively, then $(a,p)=(p_1 a_1 + p_2 a_2, p_1 + p_2)$ are the coordinates for transition t_3 (note that a is actually the mean of the distribution calculated in [14]).

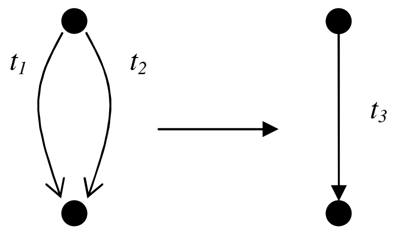


Figure 15. Alternate reduction

Cycle Reduction (t) **:** The sequential reduction step might create a state with a transition to itself. Therefore a cycle reduction step may be necessary. For this step, we assume that transition t is internal, in other words $t.e$ is not visible to the users, or is visible but of no importance to our performance study (for example, if we are interested in MTTF then the only transition we would like to monitor is the fail transition, and all other transitions could be considered as internal or not visible). Given a cycle transition t, and q non-cyclic transitions ($q \in \mathrm{N}$, the set of positive integers) $t_1, t_2, \ldots, t_q$, out of the same state (refer to Figure 16), we would like to eliminate transition t. Assume that (a, p), (a_1, p_1), $(a_2, p_2), \ldots$, (a_q, p_q) are the coordinates for the transitions t, t_1, $t_2, \ldots, t_q$ respectively. To eliminate the self-cycle, we propose the following steps:

1. Change the coordinates of transitions t_1, $t_2, \ldots, t_q$ so as to reflect the average time spent going around the cycle. Since the probability of doing the cycle is p, then the average time spent doing the cycle over and over is $0p^0 + ap^1 + 2ap^2 + \ldots. = a\sum_{i=1}^{\infty} ip^i = a\frac{p}{(1-p)^2}$, (for more information, check [14]).

2. Renormalize the probabilities for transitions t_1, $t_2, \ldots, t_q$. This is done by dividing their probabilities by $1-p$.

So $(a',p') = (a_i + a\frac{p}{(1-p)^2}, \frac{p_i}{1-p})$ becomes the new coordinate for transition t_i, $i \in 1,...,q$.

We note that *a'* is actually the mean of the distribution calculated in [14].

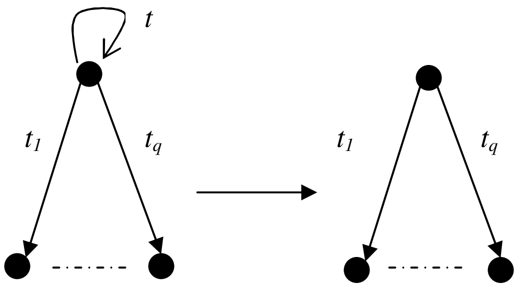


Figure 16. Cycle removal

In the next subsection, we present an algorithm to remove a state from an SMP while preserving the mean passage time equivalence over the non-deleted states. The algorithm removes one state at a time, in other words, given an $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ and a set of states M such that the set of states $S - M$ are to be deleted. Then we use the algorithm below to delete the states in $S - M$ one by one. It is not difficult to note that the order of state deletion does not affect the final SMP *G'*.(refer to [14])

Note that the algorithm is intended to delete states according to the user specification, in other words, the user specifies the states that are not of interest to a particular performance study, then these states are deleted. This will be explained in more details in Chapter 7.

6.3.2. Algorithm

The algorithm takes as input an SMP *G* and a state *s* to be deleted. It outputs an SMP *G'*.

Variables: Φ is a set of paths of length 2, Ψ is a set of transitions.

Algorithm:

```
    $\Psi = \phi$;

    $\Phi$ = the set of paths of length 2 that have s as the middle state;

    STEP1:

    Repeat for each path $t_1 t_2 \in \Phi$

        combine the two transitions $t_1$ and $t_2$ to form one transition $t$ using the sequential
        reduction;

        $\Phi = \Phi - \{t_1 t_2\}$;

        $\Psi = \Psi \cup \{t\}$;

    Until $\Phi = \phi$.

    STEP2:

    While there are two alternate transitions $t_1, t_2 \in \Psi$ with $t_1.e = t_2.e$

        combine the two transitions $t_1$ and $t_2$ to form one transition $t$ using the branch
        reduction;

        $\Psi = [\Psi \cup \{t\}] - \{t_1, t_2\}$;

    STEP3:

    While there is a self loop $t$ in $\Psi$ with an event $t.e$ tat is considered internal

        delete the self loop using the cycle reduction step;

        $\Psi = [\Psi - \{t\}]$;

    Output the resulting SMP
```

End Algorithm. *** should not be at beginning of page

Theorem 6.1. Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an SMP and let $G' = (S - \{s\}, s_0, \mathrm{E}, F', A, \mapsto', K)$ be the SMP obtained by applying the algorithm above to a state $s \in S$, $s \in S$, let $S' = \{s_1, ..., s_m\}$ be the set of states $\in S$ that are directly connected to s in G then

$G \overset{S-\{s\},S-\{s\}}{\equiv} G'$, and

$G \overset{S-S',S-S'}{\cong} G'$

Proof. Straightforward.

∎

6.4. Complexity

We have defined an equivalence over semi-Markov processes which is based on state simplification. The time complexity to delete one state in the SMP is $(n-1)^2 c_1 + [(n-1)(n-2) - x]c_2 + xc_3$ in the worst case, i.e. if the SMP is heavily connected, where x is the number of cycles generated from the sequential reduction step (Step1) and:

- $(n-1)^2 c_1$ is the time required to do Step 1 of the algorithm with c_1 being the time needed to calculate the mean and probability of a newly formed transition (i.e. the time needed to calculate (*a,p*)).

- $[(n-1)(n-2) - x]c_2$ is the time required to do Step 2 of the algorithm with $[(n-1)(n-2) - x]$ being the number of alternate transitions and c_2 being the time needed to calculate the mean and probability of a newly formed transition.

- xc_3 is the time required to do Step 3 of the algorithm, with x being the number of self-loops and c_3 being the time needed to calculate the mean and probability of a newly formed transition.

If we assume that $c_1 = c_2 = c_3$ then algorithm complexity becomes: $c_1[(n-1)^2 + (n-1)(n-2)]$

Note that, contrary to [14], the time to calculate the average distribution and probability of the newly created transitions, i.e. c_1 and c_2 and c_3 is negligible compared to the time needed to get the actual time distributions for these transitions since the latter involves integral calculations. So the overall complexity is $O(n)^2$

6.5. Effects on the Original NRGSMP

Let $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ be an NRGSMP, and let $G'' = (S', s'_0, \mathrm{E}', F'', \mapsto', K'')$ be the SMP obtained from G following Algorithms 1 and 2 of Chapter 4. Then we have that G'' s-simulates G. Let $\Delta = \{\Delta_j, j \in J\}$ and $R : S \to \Delta$ be the partition of states ($S' = \Delta$) and the correspondence that together establish the transient state simulation. Let $s \in S$, and let $N = (S' - R(s), s'_0, \mathrm{E}', F'', \mapsto' / S' - R(s), K'' / S' - R(s))$ be the SMP obtained from G'' by deleting from S' all states in the set $R(s)$ following the above algorithm. let $S' = \{s_1, ..., s_m\}$ be the set of states $\in S$ that lead directly to s in G. Then we have the following lemma:

Lemma 6.1. Let $s' \in S - S'$, then $\pi^G(s') = \sum_{r \in R(s')} \pi^{G''}(r) = \sum_{r \in R(s')} \pi^N(r)$

Proof. Straightforward.

∎

So, to facilitate performance analysis for NRGSMPs, we identify in G the set Φ of all states that are not key to our performance analysis and that are not directly accessible form our states of interest. Then we delete from G'' all states in $R(\Phi)$, where R is the relation described in the lemma above, and we do our analysis on the reduced SMP.

The performance measures in G that could be deduced from the reduced SMP are all the measures that depend on the steady state probability of some of the states such as probability of failure.

Chapter 7: Illustrations and Applications

In this Chapter, we discuss the issue of how to check whether a GSMP satisfies the properties of an NRGSMP. Then, we present a case study illustrating the transformation presented in Chapter 4. However, we start first with a simple example to show the reader how the different chapters fit together.

7.1. A Simple Example

In this section, we present to the reader the big picture: how everything fits together. We introduce an example of an NRGSMP, and explain the performance measures that we need to extract from the process. Then we apply the simplification procedure presented in Chapter 5 to the states in the NRGSMP that satisfy the conditions stated in Chapter 5. The next step would be to transform the simplified NRGSMP into an SMP following the algorithms presented in Chapter 4. Then, the resulting SMP is simplified by deleting some of its states following the algorithm presented in Chapter 6.

We consider the model of a machine that receives requests and services them over and over again. The machine can service one request at a time, and requests are generated when the machine is not in service. The request generation is modeled by event r. Request servicing is a two phase procedure, in the first phase the service is done by a component of the machine that is failure prone, however, the second phase of service is assumed never to fail. The service is modeled by the consecutive events s_1 and s_2, both have a generally distributed lifetime duration. The machine keeps working for a constant period of time and then undergoes tune-up. If the machine is servicing a request when tune-up is due, the machine aborts the current service to undergo the tune-up. The interval between two consecutive tune-ups is constant and is modeled by event a. The tune-up process is generally distributed and is modeled by event u. In the first service phase, the machine can fail, at that time it has to undergo repair, the failure and repair are modeled by events f and p, respectively; they both have generally distributed lifetimes. If the machine fails and is repaired, it reinitializes event a (note that a might have a different lifetime distribution in state 4 and 0, because the tune-up time after a repair is not so urgent). The model is depicted in Figure 17. Transition d is immediate.

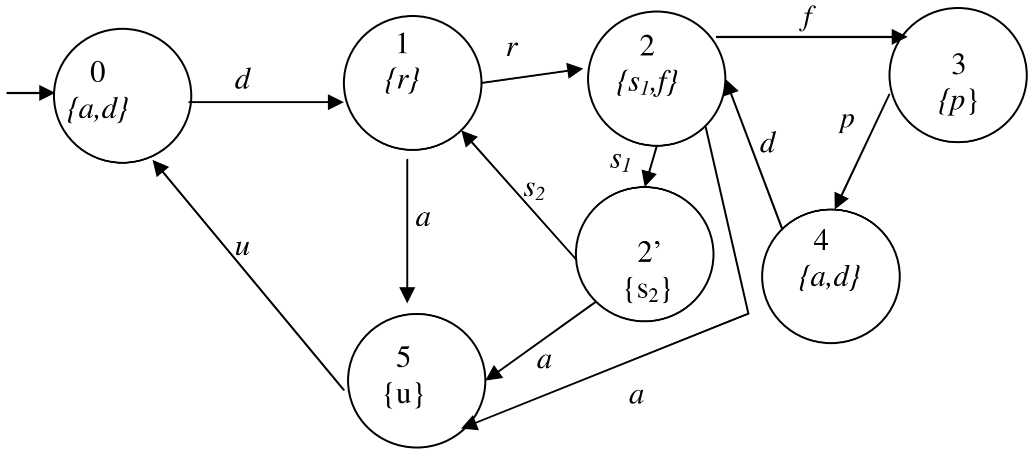


Figure 17. Example of an NRGSMP

We assume that the measure that we are interested in is probability of failure, so the states that should not be deleted are the fail state, the starting state, and state 4, i.e. states in the set $\Sigma = \{0,3,4\}$

7.1.1. NRGSMP Simplification

As explained in the previous section, we need to delete all states apart from states in the set $\Sigma = \{0, 3,4\}$. Figure 18 shows the only ESMP in the NRGSMP G of Figure 17.

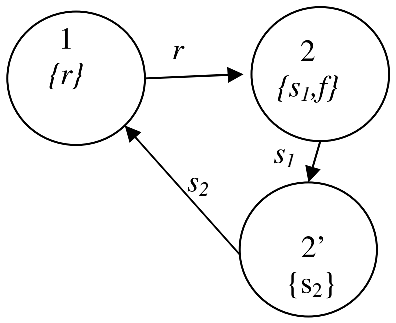


Figure 18. ESMP

Note that the only state that satisfies conditions 1, 2, 3, and 4 of Section 5.3 is State 2'. Therefore, this is the only state we can delete using the method described in Chapter 5. To delete State 2', we aggregate transitions: $2 \xrightarrow{s_1} 2'$ and $2' \xrightarrow{s_2} 1$ to form a single transition $2 \xrightarrow{s} 1$ where $s = s_1 s_2$, and the distribution associated with event s is the convolution of the distributions associated with events s_1 and s_2. The resulting NRGSMP is shown in Figure 3.

7.1.2. NRGSMP to HMRP and then to SMP

The next step would be to transform the simplified NRGSMP G' into an HMRP and then into an SMP G''. The transformations were discussed in Chapter 4, the resulting SMP is shown in Figure 9, and the distributions can be calculated following Algorithm 2 of Chapter 4. For the remaining of this chapter, we assume that R is the relation that establishes the s-simulation between G' and G'' and we denote by m_e^i the mean time of the distribution of an event e in state *i* and p_e^i the probability that the transition governed by event e occurs out of state *i*. We omit *i* whenever the state we are referring to is obvious.

7.1.3. SMP Simplification

As discussed earlier, the measure of interest in this chapter is the probability of failure, so the events that are of interest to us are the ones associated with failure i.e. *f*. All other events are not important and can be assumed to be internal or not visible to the users, usually denoted by τ. The states we identified to be of interest in G' are $\Sigma = \{\ 0,3,4\}$, so for G'', the states of interest would then be $R(\Sigma) = \{0-0, 3-3, 5-4\}$, so all the states outside this set, i.e. the states $\{1-1, 2-2, 4-5, 6-2, 7-1\}$ can be deleted. We will illustrate the SMP simplification by deleting one of the states in the set $\{1-1, 2-2, 4-5, 6-2, 7-1\}$ which is state 1-1. For that purpose we consider the sub-SMP that contains the states that are directly connected to state 1-1, the sub-SMP that we consider is shown Figure 19. Note that we renamed event *a* out of state 2-2 as *a'* so that each transition would have a unique event name, this makes the presentation easier.

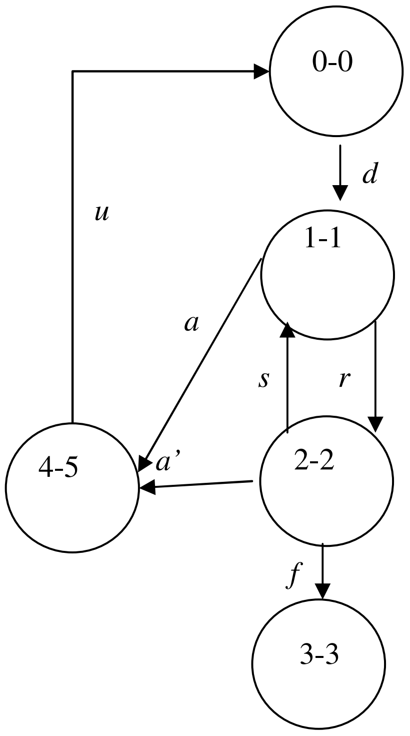


Figure 19. The sub-SMP

As mentioned earlier, all events aside from *f* are internal transitions (τ), however, in what follows, we choose to show the events so that the reader keeps track of the simplifications done and understands what each transition stands for. For that reason, we use the following notation:

- ee': stands for the event on the transition resulting from the sequential reduction of two transition whose events are e and e' respectively.
- $e+e'$:stands for the event on the transition resulting from the alternate reduction of two transition whose events are e and e' respectively.

- $(e)^{e'}$: given a transition *t* out of a state say s, with event e, and a self cycle *t'* (from s to itself) whose event is e', then we denote by $(e)^{e'}$ the new event on transition *t* obtained after the removal of the self cycle.

Figure 20 shows two steps in the deletion of state 1-1. Figure 20(a), is obtained after performing Step1 of the algorithm in Section 6.3.2. Transition $(sa+a')^{sr}$ is the result of merging the alternate transitions: sa and a' following by the removal on the self cycle in state 2-2.

To illustrate, we will calculate the mean time of the distributions and the transition probabilities for Figure 20 in terms of the previous ones (Figure 19):

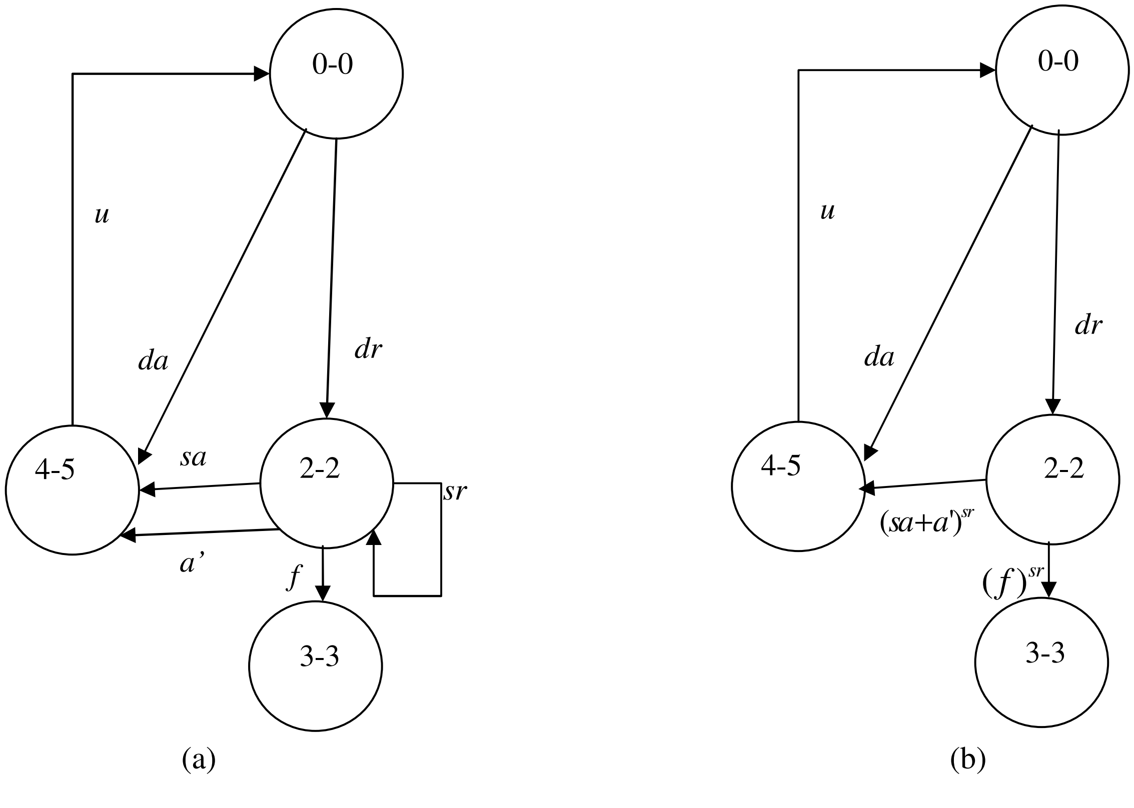


Figure 20. Deletion of State 1-1

In Figure 20 (a):

$m_{dr} = m_d + m_r$, $P_{dr} = P_d \times P_r$ and

$m_{da} = m_d + m_a$, $P_{da} = P_d \times P_a$ and

$m_{sa} = m_s + m_a$, $P_{sa} = P_s \times P_a$ and

$m_{sr} = m_s + m_r$, $P_{sr} = P_s \times P_r$

In Figure 20 (b):

$m_{(sa+a')^{sr}} = (P_{sa}m_{sa} + P_{a'}m_{a'}) + m_{sr}\dfrac{P_{sr}}{(1-P_{sr})^2}$, $P_{(sa+a')^{sr}} = \dfrac{P_{a'} + P_{sa}}{1-P_{sr}}$ note that to obtain these formulas, we need to combine transitions a' and sa using alternate reduction, the coordinates of the new transition are: $(P_{sa}m_{sa} + P_{a'}m_{a'}, P_{a'} + P_{sa})$ then we apply the cycle removal step to obtain the coordinates for $(sa+a')^{sr}$ above.

The cycle removal changes the coordinated for f as well as follows:

$m_{f^{sr}} = m_f + m_{sr}\dfrac{P_{sr}}{(1-P_{sr})^2}$, $P_{f^{sr}} = \dfrac{P_f}{1-P_{sr}}$

7.2. Properties of NRGSMPs

To check whether a GSMP $G = (S, s_0, \mathrm{E}, F, A, \mapsto, K)$ is an NRGSMP, we need to find all the cycles in the GSMP and check whether each of the cycles satisfies the conditions set on the cycles of an NRGSMP. Checking the type of the cycles could be done in parallel with finding the cycles by recording the value of $A(s) - K(s)$ for all states on the cycle and checking whether every transition $s \xrightarrow{e} s'$ on the cycle satisfies the condition $e \in K(s)$. So the problem boils down to finding all the cycles in the NRGSMP.

A number of algorithms for finding the cycles of a directed graph are based on the backtracking strategy [81]. These algorithms are bounded by $O(|S| + E(C+1))$, where $|S|$ is the number of vertices (states) in the graph, E is the number of edges ($E = |\mapsto|$), and C is the number of cycles in the graph.

In [66] the authors designed a novel algorithm to find the cycles in a directed graph, the algorithm is bounded by $O(E)$ which is a big improvement from the previously known algorithms.

In the next section, we will present a class of systems known as software rejuvenation models.

7.3. Case Study

7.3.1. Software Rejuvenation

Many software systems run for long periods of time, "some of the faults causes them to age due to the error conditions that accrue with time and load" [80]. These accumulated error conditions cause degradation in the system's performance and eventually lead to failure [80]. In [74] the authors cite the example of the progressive depletion of the operating system's resources such as *free memory available* due to software errors such as "memory leaks and incomplete cleanup of resources after use". For more examples, the reader is referred to [1],[5].

A well-known preventative approach to counteract software aging is referred to as software rejuvenation. Software rejuvenation is a low cost approach that involves stopping the running system periodically and restarting it after cleaning its internal state [74].

Cleaning the internal state involves several approaches such as: garbage collection, re-initialization of internal data structures, and flushing the internal tables of the operating system [80].

An optimal schedule for software rejuvenation increases the system's availability by reducing the system's failure, and hence reduces the amount of repairs needed. However, setting an optimal schedule is not an easy task [74]. To be able to set an optimal schedule, one needs to define a good model for the system that adequately represents the systems components and failure rates.

Several models have been proposed in literature to find an optimal rejuvenation schedule. In [57], Huang et al. describe a basic model in which the degradation of the system is a two step process. Originally the system is in a clean state where no failure is possible, then, after a random amount of time, the system moves to a failure-prone state. From that state two actions are possible: a complete failure with return to the clean state after repair, or rejuvenation with return to the clean state. In [37], the process is modeled as a CTMC and its steady state availability is calculated. In [36], Dohi et al. extend this basic model by assigning any type of distributions to the events, thus making the model an SMP. In [35], Dohi et al. use a modification of the basic model: after repair from failure, the system moves to the rejuvenation state. The reason behind the modification is that after failure, a system would be restarted and cleaned. Analysis of the availability of the modified semi-Markov model is given in [35]. In [39], Garg et al. introduced the concept of periodic rejuvenation into the basic model (with deterministic interval between successive rejuvenation). The new model is represented using a Markov regenerative process satisfying the enabling restriction. The model is shown in Figure 21 where

- State 0 is the clean state
- State 1 is the failure prone state
- State 2 and State 4 are the rejuvenation states
- State 3 is the failure state.

- The distribution associated with event *m* represents the interval between successive rejuvenations, and the occurrence of event *m* indicates the end of that interval. The rejuvenation and the repair procedures are both modeled by the same event label *r*: note, however, that repair and rejuvenation need not have the same time distribution associated with them; in other words, $F_2(r) \neq F_3(r)$. The aging of the system is modeled by event *a*, and the failure of the system is modeled by event *f*. We assume that the rejuvenation after failure takes a longer duration.

- Note that the only non-regenerative state is State 1 as the time until rejuvenation is initialized in State 0.

In all the rejuvenation models cited above, it is assumed that state change from the clean state to the failure prone state is observable [69]. Given this assumption, it makes sense to assume that the user would be able to observe the cause of deterioration of the system in the failure prone state. Such observation leads to more efficiency during the repair phase. And both assumptions justify the need to model the aging of the system and the failure of the system as two separate events.

The model was analyzed in [39] using the method of Markov regenerative processes. Then the same model, with no restrictions on the distributions governing its events (i.e. not satisfying the enabling restriction) was analyzed in [35],[36] by transforming it into a 3 state semi-Markov process by merging states 0 and 1 and also states 4 and 2. The disadvantages of this method is that some measures of interest can not be computed from the reduced semi-Markov process such as the proportion of time the state is in the failure prone state versus the clean state: $\frac{\pi_1}{\pi_0}$.

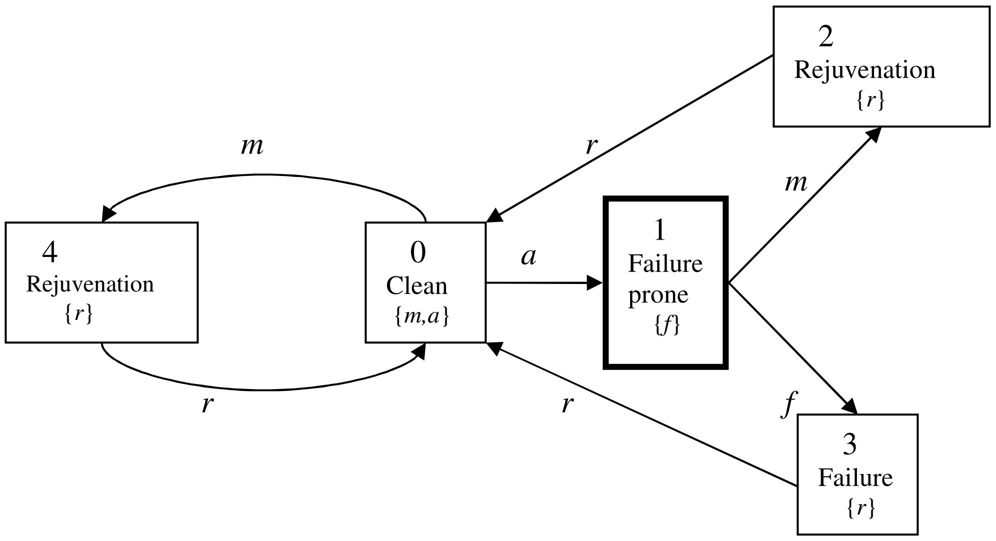


Figure 21. Periodic rejuvenation

In the subsection 7.3.3, we will extend the periodic rejuvenation model presented above to a random rejuvenation model, i.e. to a model where the time until rejuvenation is a random variable that is not periodic, and then we will apply the method presented in Chapter 4 of the thesis on the new model, and calculate its steady state probability. Moreover, in the next sub-section, 7.3.2, we will extend the periodic rejuvenation model by considering a periodic rejuvenation of a system with a backup unit.

7.3.2. Extension of the Periodic Rejuvenation Model

In this sub-section, we consider the periodic rejuvenation of a system that has a backup unit. We assume that rejuvenation includes the process of preventative maintenance, i.e. replacing any worn-out parts.

In our model, the main system is rejuvenated periodically and after a failure. Once in rejuvenation, the backup unit replaces the main unit. The backup unit undergoes rejuvenation after each failure and minor rejuvenation every time it replaces the main unit for a period of time. Minor rejuvenation is assumed to be an immediate action and is not modeled for that reason (otherwise, one could assume that the aging of the main unit is a longer procedure than the minor rejuvenation). We assume that we have one rejuvenation facility, in other words, if both units are due for rejuvenation, then the rejuvenation has to be done sequentially, one after the other.

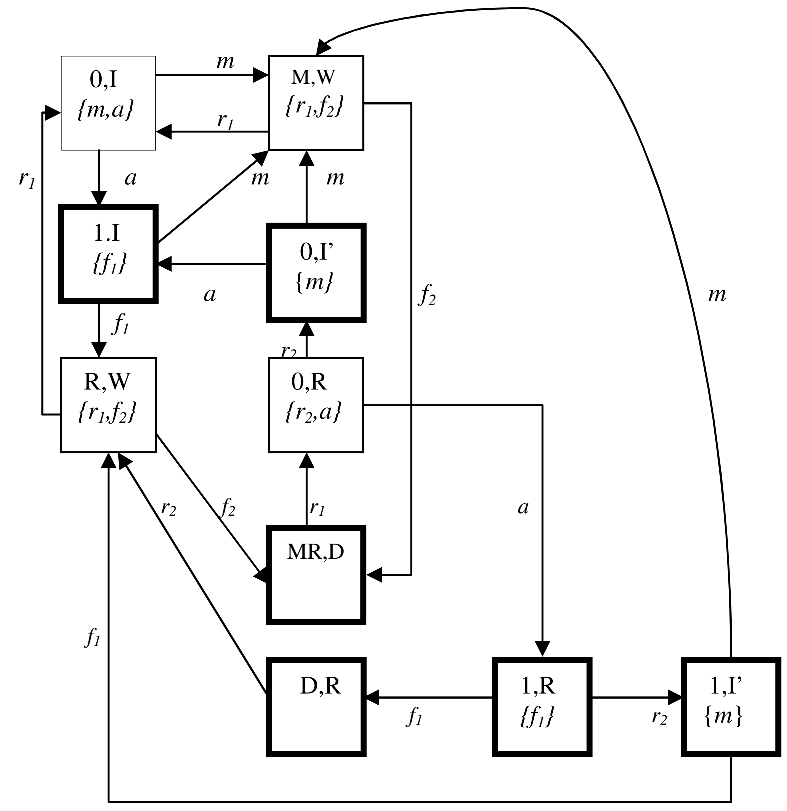


Figure 22. Periodic rejuvenation with backup unit

Originally, the main unit is in clean state where no failure is possible and the backup unit is idle, then, after a random amount of time the main unit moves to a failure prone state.

If failure occurs from that state, then the main unit undergoes rejuvenation and the backup unit starts working. While in working state, the backup unit could fail, once failed the backup unit moves to a down state and waits until the rejuvenation facility becomes available. The backup unit could be in three modes: idle (I), working (W), down (D), or under rejuvenation (R). Periodic rejuvenation and rejuvenation after failure are modeled by event *r*, however, the distribution associated with the rejuvenation after a failure is different than that associated with the periodic rejuvenation (because of the repair component). Note that, the most undesirable state is when the system is under rejuvenation and the backup unit fails, in that case, both units would not be working. As soon as one of the units is rejuvenated, the system starts running again.

The model is shown in Figure 22, the labels inside every state are composed of two letters separated by a coma, the first indicate the status of the main system and the second indicates the status of the backup unit:

- I, W, D, M and R : stand for idle, working, down, rejuvenation and repair respectively. MR stands for rejuvenation or repair.
- 0 stands for clean state and 1 stands for failure prone state.
- The distribution associated with event *m* represents the interval between successive rejuvenations, and the occurrence of event *m* indicates the end of that interval. The same event labels r_1, and r_2 represent the rejuvenation and the repair procedures for the main unit and the backup unit, respectively. Note however that repair and rejuvenation need not have the same time distribution associated with them. The aging of the main unit is modeled by event *a*, and the failure of the main unit and the backup unit is modeled by events f_1 and f_2, respectively.

Note that states 1,I; 1I'; MR,D; 0,I; I,R; and D,R are the non-regenerative states. Moreover, all the cycles in the model are regenerative cycles; hence the above model is an NRGSMP. We will demonstrate in sub-section 7.3.4 how to transform the NRGSMP into an HMRP.

7.3.3. Analysis of the Periodic Rejuvenation Model

Here we consider the example in Figure 21 above. As discussed before, the process is an HMRP with one non-regenerative state: State 1. To transform the process into an SMP, we need to find $Av\,\mathrm{Re}\,s(m,1)(x)$. Then we can find the steady state probability of the resulting SMP by calculating the SSP of the embedded Markov chain and the mean time spent in each state of the SMP (refer to Chapter 2).

We denote by $F_i(e)$ the distribution of event e from state i, and denote by $P=[P_{ij}]$ the transition probabilities for the embedded Markov chain. Then P has the following form:

$$\begin{bmatrix} 0 & P_{01} & 0 & 0 & 1-P_{01} \\ 0 & 0 & P_{12} & 1-P_{12} & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where $P_{01}=\int_0^\infty \frac{dF_0(a)(x)}{dx}(1-F_0(m)(x))dx$ and $P_{12}=\int_0^\infty \frac{dF_1(m)(x)}{dx}(1-F_1(f)(x))dx$, (note that $F_1(m)(x)=Av\,\mathrm{Re}\,s(m,1)(x)$).

Now, $Av\,\mathrm{Re}\,s(m,1)(x)=Av\,\mathrm{Re}\,s(m,1,0\xrightarrow{a}1)(x)$, hence, from Theorem 4.6, we have that

$$Av\,\mathrm{Re}\,s(m,1)(x)=\int_0^\infty \frac{F_0(m)(x+x')-F_0(m)(x')}{1-F_0(m)(x')}\,\frac{\frac{dF_0(a)(x')}{dx'}(1-F_0(m)(x'))}{\int_0^\infty \frac{dF_0(a)(x'')}{dx''}(1-F_0(m)(x''))dx''}dx'=$$

$$\int_0^\infty \frac{[F_0(m)(x+x')-F_0(m)(x')]\dfrac{dF_0(a)(x')}{dx'}}{P_{01}}dx'$$

The steady state probability vector of the embedded Markov chain: $\pi=[\pi_0,\pi_1,\pi_2,\pi_3,\pi_4]$ satisfies the following equations:

$\pi_j=\sum_{i=0}^{4}\pi_i P_{ij}$ for $j\in\{0,1,2,3,4\}$ and $\sum_{i=0}^{4}\pi_i=1$, which yields the following: $\pi=\left[\pi_0,\ P_{01}\pi_0,\ P_{01}P_{12}\pi_0,\ P_{01}(1-P_{12})\pi_0,\ (1-P_{01})\pi_0\right]$. Applying $\sum_{i=0}^{4}\pi_i=1$, we get $\pi_0=\dfrac{1}{2+P_{01}}$ hence:

$$\pi=\left[\frac{1}{2+P_{01}},\ P_{01}\frac{1}{2+P_{01}},\ P_{01}P_{12}\frac{1}{2+P_{01}},\ P_{01}(1-P_{12})\frac{1}{2+P_{01}},\ (1-P_{01})\frac{1}{2+P_{01}}\right]$$

To calculate the steady state probability of the SMP, it remains to find the mean waiting time in every state of the process denoted by M_i, $i\in\{0,1,2,3,4\}$. Recall from Section 2.3.2.1 that $M_i=\sum_{j=0}^{4}P_{ij}E(T_{ij})$, (recall that $E(T_{ij})$ is the expected sojourn time in state i knowing that state j will be visited next) hence we get the following equations:

$$M_0=P_{01}E(F_0(a))+P_{04}E(F_0(m))$$

$$M_1=P_{12}E(F_1(m))+P_{13}E(F_1(f))$$

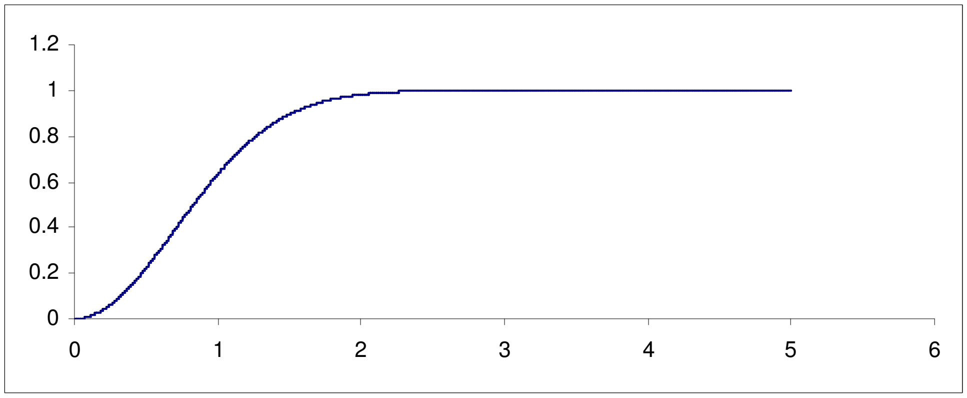
$$M_2=E(F_2(r))$$

$$M_3=E(F_3(c))$$

$$M_4=E(F_4(r))$$

In rejuvenation studies, the aim is usually to determine the best rejuvenation schedule, and that could be done by testing different distributions and checking which one is the best in terms of minimizing the probability of failure of the system and the overhead caused by rejuvenation. The probability of being in the fail state F, is calculated as follows: $F = \frac{M_3\pi_3}{\sum_{i=0}^{4} M_i\pi_i}$. And the overhead caused by the rejuvenation could be be minimized by minimizing the expected instantaneous rejuvenation cost in steady state: C, which can be calculated as follows: $C = \frac{c(M_4\pi_4 + M_2\pi_2)}{\sum_{i=0}^{4} M_i\pi_i}$, where c is the cost of rejuvenation per unit of time.

The above analysis applies to any type of distribution. We chose to illustrate these results with the following distributions: The aging of the system modeled by event a, it is assumed to have the Weibull distribution. The Weibull distribution is often used in the field of life analysis to model the aging of a system. The Weibull distribution is characterized by a parameter k which represents the aging rate. We assume here that $k = 2$. The distribution is given by: $F_0(a)(x) = 1 - e^{-x^2}$, and the distribution is represented in the figure below



As mentioned before, to be able to determine the best rejuvenation schedule, we need to test different distributions and to check which one is the best in terms of minimizing the failure of the system and the overhead caused by rejuvenation. In this example, we will give rejuvenation the same distribution as the aging which is the Weibull distribution with parameter $k = 2$. So $F_0(m)(x) = 1 - e^{-x^2}$, and calculate the effect of such a schedule on the probability of failure in the system as well the expected rejuvenation cost.

The failure of the system is modeled by event f. Once we reach State 1, the system fails after a fixed period of time. $F_1(f)(x) = \begin{cases} 1 & x \geq 1 \\ 0 & x < 1 \end{cases}$.

$F_2(r)(x)$, $F_4(r)(x)$, and $F_3(r)(x)$ have means equal to $\frac{1}{90}$, $\frac{1}{90}$ and $\frac{1}{60}$ respectively, meaning that the periodic rejuvenation has a mean of $\frac{1}{3}$ of a day, and the rejuvenation after failure has a mean of $\frac{1}{2}$ a day (note from the steady state formulas above that these distributions affect the result only through their means).

With the above distributions, we get the following:

$P_{01} = \frac{1}{2}$, and hence $P_{04} = \frac{1}{2}$

$$F_1(m)(x) = Av\operatorname{Re}s(m,1)(x) = \int_0^\infty x' e^{-x'^2}(e^{-x^2} - e^{-(x'+x)^2})dx' = \frac{1}{2} - \int_0^\infty x' e^{-x'^2-(x+x')^2}dx'$$

$P_{13} = 1 - F_1(m)(1) = \frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2}dx'$, and hence $P_{12} = \frac{1}{2} - \int_0^\infty x' e^{-x'^2-(1+x')^2}dx'$

$$\pi = \left[\frac{2}{5},\ \frac{1}{5},\ \frac{\frac{1}{2} - \int_0^\infty x' e^{-x'^2-(1+x')^2}dx'}{5},\ \frac{\frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2}dx'}{5},\ \frac{1}{5}\right]$$

Also, the mean waiting times in every state can be calculated as follows:

$$M_0 = P_{01}E(F_0(a)) + P_{04}E(F_0(m)) \ = \int_0^\infty x'^{\frac{1}{2}} e^{-x'} dx'$$

$$M_1 = P_{12}E(F_1(m)) + P_{13}E(F_1(f)) = \frac{1}{2} - \int_0^\infty x' e^{-x'^2-(1+x')^2} dx' + \frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2} dx'$$

$$M_2 = \frac{1}{90},\ M_3 = \frac{1}{60},\ M_4 = \frac{1}{90}$$

Finally, the probability of being in the fail state F is calculated as follows:

Now let $\Gamma = \sum_{i=0}^{4} M_i \pi_i$, then

$$\Gamma = \frac{2}{5}\int_0^\infty x'^{\frac{1}{2}} e^{-x'} dx' + \frac{1}{5}\frac{1}{2} - \int_0^\infty x' e^{-x'^2-(1+x')^2} dx' + \frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2} dx' +$$

$$\frac{\frac{1}{2} - \int_0^\infty x' e^{-x'^2-(1+x')^2} dx'}{450} + \frac{\frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2} dx'}{300} + \frac{1}{450}.$$

And

$$F = \frac{M_3\pi_3}{\sum_{i=0}^{4} M_i \pi_i} = \frac{\frac{1}{2} + \int_0^\infty x' e^{-x'^2-(1+x')^2} dx'}{300\Gamma}$$

Moreover, the expected instantaneous rejuvenation cost in steady state $C = \frac{c(M_4\pi_4 + M_2\pi_2)}{\sum_{i=0}^{4} M_i \pi_i} = \frac{2c}{135000\Gamma}$.

7.3.4. Analysis of the Periodic Rejuvenation with Back-up Model

To understand how our NRGSMP model will be transformed into an HMRP, we need to identify, for each non-regenerative state s, the set of single regenerative states Γ_s that lead to it. Then we need to divide the set Γ_s into subsets of traces $\{\delta_s^1,...,\delta_s^m\}$ such that each subset δ_s^i, is a single-*AvRes* set. Recall that, the number of subsets obtained, m, is equal to the number of copies of the non-regenerative state s in the HMRP (as explained in Chapter 4). Among the states in Figure 22, the only states that have more than one single regenerative state leading to them are state 1,I and state MR,D. These traces are: $\Gamma_{1,I}=\{(0,I\xrightarrow{a}1,I)$, $(0,R\xrightarrow{r_2}0,I'\xrightarrow{a}1,I)\}$ and $\Gamma_{MR,D}=\{(R,W\xrightarrow{f_2}MR,D)$, $(M,W\xrightarrow{f_2}MR,D)\}$, (note that the sets $\Gamma_{MR,D}$ and $\Gamma_{1,I}$ are finite because we do not have near semi-Markovian cycles in the NRGSMP). Now each of trace in $\Gamma_{1,I}$ is a single-*AvRes* set, and similarly, each trace in $\Gamma_{MR,D}$ is a single-*AvRes* set. Hence to transform the model into an HMRP, we need to create two copies of state 1,I one accessible through trace $0,I\xrightarrow{a}1,I$ and the other accessible through trace $0,R\xrightarrow{r_2}0,I'\xrightarrow{a}1,I$. Similarly, we need to create two copies of state MR,D one accessible through trace $R,W\xrightarrow{f_2}MR,D$ and the other accessible through trace $M,W\xrightarrow{f_2}MR,D$. The HMRP is shown in Figure 23.

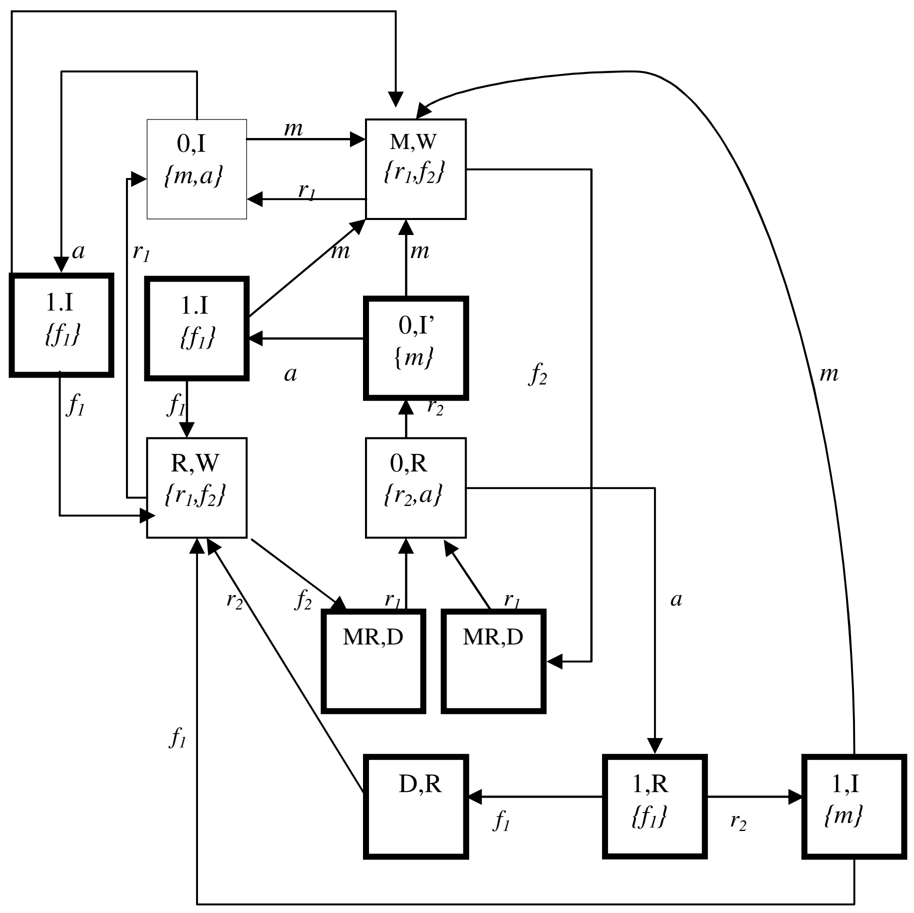


Figure 23. HMRP for the periodic rejuvenation with back-up.

To transform the non-regenerative states in the HMRP into regenerative we need to find the average residual distributions of the active events in the non-regenerative state as was illustrated in the previous sub-section.

Chapter 8: Conclusion and Future Directions

In the literature, there are two main methods that attempt to analyze GSMPs: the regenerative and the supplementary variable methods. Both methods can be applied to a subset of GSMPs, those that implement the “enabling restriction” meaning that only one non-exponentially distributed clock can be active at a given time. Imposing this restriction leads to algorithms with reasonable costs. Going beyond the restriction is one of the most challenging open issues in the field. Other methods exist that deal with GSMPs with special type of distributions such as the continuous phase type distribution or deterministic event durations; for such GSMPs, efficient numerical analysis can be found [8],[42].

The contributions of this thesis are the following:

- We extended the class of solvable GSMPs by allowing several generally distributed events to be enabled at any time. However, we imposed the restriction that every cycle $C = s_1 \xrightarrow{e_1} s_2 ... \xrightarrow{e_{n-1}} s_n \xrightarrow{e_n} s_1$ in the GSMP must either be *near semi-Markovian* (NSM) or *regenerative* (REG) (see

Definition 4.3). GSMPs whose cycles are either NSM or REG are referred to as near-regenerative generalized semi-Markov processes, NRGSMP, the concepts of NRGSMP is introduced in Chapter 4 of the thesis. NRGSMPs are more general than the GSMP's implementing the enabling restriction (EGSMPs). In fact, among other restrictions, the only cycles allowed in a EGSMP are either regenerative or near-Markovian. However, as discussed in Chapter 7, an important class of GSMP's is not covered by NRGSMPs; examples are the queuing networks G/G/1 of size $n \geq 3$. The method presented to solve the steady state probabilities for NRGSMPs consists of an algorithm that transforms the NRGSMP into a semi-Markov process (SMP) while preserving *steady-state simulation*, a simulation that enables us to determine the steady state probability of the NRGSMP from that of the SMP constructed. The time and space complexities of the algorithm presented in this thesis are exponential in the number of states in the set S^{rem}, i.e. states that are neither regenerative nor belong to an embedded semi-Markov process (ESMP, where an ESMP is a sub-process which is a semi-Markov process), and in the number of strongly connected ESMPs $(q-h)$, (i.e. the number of embedded strongly connected semi-Markov processes) that have at least one non-regenerative in-border (Definition 4.5). But, when applied to GSMPs whose states either belong to an ESMP with regenerative in-border or are regenerative, the algorithm becomes $O(|S|^R|S-S^R|)$ in space complexity, where S^R is the set of regenerative states, and $O(2cm^3+t|S|^2+2c|S|^2)$ in time complexity, where m is the maximum number of states of the ESMPs and t is the branching factor of the NRGSMP, (refer to Section 4.5). The regenerative method has $O(|S|^2)$ space complexity and $O(|S|^4)$ time complexity [45]. The method of supplementary variable has approximately $O(q^g|S|^2)$ time complexity and $O(|S^E|+c\sum_{g\in T^G}|S^g|+\sum_{g\in T^G}|S^g|^2)$ space complexity where S^E is the set of states in which only exponential transitions are enabled, S^g is the set of states in which the non-exponential transition g is enabled, and c denotes the time for integral calculation [45]. The exponential factor in the complexity of our algorithm limits its real applicability to the subset of GSMPs with a small number of states in the set S^{rem}, and a small value of $q-h$, However, this subset contains all GSMPs satisfying the enabling restriction.

- The method described in the point above could generate semi-Markov processes with big state spaces. For that reason, we introduced a method to remove states from the original NRGSMP while preserving the distribution of time needed to travel between non-deleted states and also preserving the transient state probabilities for a subset of the states. This method works on any GSMP and works by deleting states that, among other restrictions, belong to an embedded semi-Markov process M . The time required to delete one state $s \in S_E$, (where S_E is the state space of M) from the GSMP such that $\{e_1,...,e_n\} = A(s) - K(s)$ is $cn|S_E|$ in the worst case, where c represents the complexity of integral calculation). However, this simplification of the GSMP is worth doing, since deleting one state from the biggest ESMP in the NRGSMP, leads to an SMP with as much as $t^{|S^{rem}|+(q-h)}|S^{rem}|$ fewer states.

- Another algorithm with the aim of dealing with the state space explosion of the resulting SMP was presented. This simplification technique for semi-Markov processes is based on Bradley's simplification algorithm [14], the only difference is that we are interested in preserving the mean of the passage-time distributions between non-deleted states rather than actual distributions. In fact, the technique deletes states from the SMP while preserving the average time to travel between non-deleted states, or what we call *mean passage time equivalence*. The method could delete any state in the SMP. The measures that are preserved are the performance measures that depend on the mean time to travel between the states of the SMP, such as MTTF, availability and reliability. The cost of deleting one state by following this technique is $O(|S|)^2$.

As future work, we would like to:

- Explore ways to further reduce the space complexity of the algorithm that transforms a GSMP into an SMP, and to generalize it to cover a broader subclass of GSMP's. We are currently trying to do this by generalizing the $AvRes$ set to contain structures other than paths and ESMPs. The first thing we would like to explore is to add embedded EGSMPs to the $AvRes$ set, which are sub-processes in the GSMP that, when taken as a separate entity, become GSMPs implementing the enabling restriction. This would generalize our method as it would allow more structures to be integrated in the GSMP.

- Another possibility we could explore is the generalizing of our method through the use of recursive approximation for the residual times that can not be solved analytically.

- We would like also to explore the transient-state simulation further by transforming the NRGSMP into a non-homogeneous semi-Markov process. And checking the complexity of finding the transient state probabilities of the NRGSMP from those of the resulting non-homogeneous semi-Markov process.

- Finally we would like to translate the restrictions that make an NRGSMP to the field of stochastic Petri-nets. This problem, besides having merit by itself could help us find more realistic examples of NRGSMPs.

References

[1] Adams E., Optimizing Preventative Service of the Software Products, IBM Journal of Software and Development, pages 2-14, Vol. 28 (1), 1984.

[2] Ajmone Marsan M., Balbo G., Conte G., A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems, ACM Transactions on Computer Systems (TOCS), Vol. 2, Issue 2, pages: 93-122, 1984.

[3] Ajmone Marsan M., Chiola C., On Petri Nets with Deterministic and Exponentially distributed firing Times. Lecture Notes In Computer Science; Vol. 266, Advances in Petri Nets, covering the 7th International Conference on Applications and Theory of Petri Nets, pages: 132-145, 1987.

[4] Alur R., Dill D. L., A Theory of Timed Automata, Theoretical Computer Science Vol. 126 (2), pages: 183-235, 1994.

[5] Avritzer A., Weyuker E.J., Monitoring Smoothly Degrading Systems for Increased Dependability. Empirical Software Engineering, pages 59-77, Vol. 2 (1), 1997.

[6] Balsamo S., Product Form Queuing Networks, Lecture Notes in Computer Science; Vol. 1769: Performance Evaluation: Origins and Directions, pages: 377-401, 2000.

[7] Beilner H., Mater J., Weissenberg N., Towards a Performance Modeling Environment: News on HIT, Proceedings of the 4th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Palma de Mallorca, Plenum Publishing Corporation, pages. 57-75, 1988.

[8] Bobbio A., Horvath A., Telek M., The Scale Factor: a New Degree of Freedom in Phase-Type Approximation, International Conference on Dependable Systems and Networks, Washington DC, pages. 627-636, 2002.

[9] Bobbio A., Trivedi K., An Aggregation for the Transient Analysis of Stiff Markov Chains, IEEE Transactions on Computers , vol.35, issue 9, pages.803-814, 1986.

[10] Bobbio A., Puliafito A., Telek A., Trivedi K., Recent Developments in non-Markovian Stochastic Petri Nets, Journal of Systems Circuits and Computers, Vol.8, issue.1, pages: 119-158, 1998.

[11] Bobbio A., Kulkarni V.G., Puliafito A., Telek M., Trivedi K., Preemptive Repeat Identical Transitions in Markov Regenerative Stochastic Petri Nets. Proceedings of the Sixth International Workshop on Petri Nets and Performance Models, Page: 113, 1995.

[12] Bolch G., Greiner S., de Meer H., Trivedi K., Queuing Networks and Markov Chains, John Wiley & Sons, 1998.

[13] Bradley J.T., Semi-Markov PEPA: a Contradiction in Terms?, Proceedings of 2nd International Workshop on Process Algebras and Stochastically Timed Activities, pages 1-6, 2003.

[14] Bradley J.T., A Passage-Time Preserving Equivalence for Semi-Markov Processes, Lecture Notes in Computer Science, Vol. 2324, Computer Performance Evaluation: Modelling Techniques and Tools, pages.178–187, 2002.

[15] Bradley J.T., Vowden C.J., Report on Extracting Transient Distributions from Semi-Markov Processes, Technical Report, Department of Computer Science, University of Durham, DH1 3LE, UK, 2001.

[16] Bradley J.T., Davies N., Reliable Performance Modeling with Approximate Synchronizations, Proceedings of the 7th Workshop on Process Algebras and Performance Modeling. Prensas Universitarias de Zaragoza, pages 99–118, 1999.

[17] Bravetti M., Bernardo M., Gorrieri R., Towards Performance Evaluation with General Distributions in Process Algebra, Lecture Notes in Computer Science Vol. 1466, Proceedings of the 9th International Conference on Concurrency Theory (CONCUR '98) , D. Sangiorgi and R. de Simone editors, Nice (France), pages 405-422, 1998.

[18] Bravetti M., Bernardo M., Gorrieri R., Generalized Semi-Markovian Process Algebra, Technical Report, University of Bologna, Department of Computer Science, UBLCS-97-9, Italy, 1997.

[19] Bremaud, P., Markov Chains Gibbs Fields, Monte Carlo Simulation, and Queues, Springer Verlag, 1999.

[20] Choi H., Kulkarni V.G., Trivedi K.S.: Markov Regenerative Stochastic Petri Nets, Performance Evaluation, Vol. 20 , Issue 1-3, pages: 337 - 357, 1994.

[21] Choi H., Kulkarni V.G., Trivedi K., Transient Analysis of Deterministic and Stochastic Petri Nets. Proceedings of the 14th International Conference on Application and Theory of Petri Nets. Chicago, June 1994.

[22] Clark G., Hillston J., Product Form Solution for an Insensitive Stochastic Process Algebra Structure, Performance Evaluation, vol.50 n.2-3, pages.129-151, 2002.

[23] Ciardo G., What a Structural World**,** Proceedings of the 9th international Workshop on Petri Nets and Performance Models, Germany, pages. 3-16, 2001. IEEE CS Press.

[24] Ciardo G., German R., Lindemann C., A Characterization of the Stochastic Process Underlying a Stochastic Petri Net, IEEE Transactions on Software Engineering, Vol. 20, Issue 7, pages: 506-515, 1994.

[25] Cinlar E., Introduction to Stochastic Processes, Englewood Cliffs, NJ, USA, 1975.

[26] Cox D.R., The Analysis of Non-Markovian Stochastic Processes by the Inclusion of Supplementary Variables, Proceeding of the Cambridge Philosophical Society, vol. 51, pages: 433-440, 1955.

[27] Cox D. R., Miller H.D., The Theory of Stochastic Processes, Chapman and Hall, 1965.

[28] Cumani A., Esp – A Package for the Evaluation of Stochastic Petri Nets with Phase-Type Distributed Transition Times. Proceedings of the International Workshop on Timed Petri Nets. Pages 144-151, Torino, 1985. IEEE Computer Science Press No. 674.

[29] D'Argenio P., Katoen J.P., Brinksma E., A Compositional Approach to Generalized Semi-Markov Processes, Proceedings of the 4th Int. Workshop on Discrete Event Systems (WODES'98), Cagliari, Italy, pages 391-397, 1998.

[30] D'Argenio P., Katoen J.P., Brinksma E., An Algebraic Approach to the specification of Stochastic Systems, In D. Gries and W.-P. de Roever (eds.), Proceedings of the IFIP Working Conference on Programming Concepts and Methods, PROCOMET'98, Shelter Island, New York, USA, pages 126-147. Chapman & Hall, 1998.

[31] D'Argenio P., Katoen J.P., A Theory of Stochastic Systems, Part I: Stochastic Automata. Information and Computation, volume 203(1), pages 1-38. 2005.

[32] D'Argenio P., Katoen J.P., A Theory of Stochastic Systems, Part II: Process Algebra. Information and Computation, volume 203(1), pages 39-74. 2005.

[33] D'Argenio P., Katoen J.P., Brinksma E., General Purpose Discrete Event Simulation using ♠, Editors: C. Priami, Proc. 6th Int. Workshop on Process Algebra and Performance Modeling (PAPM'98), Nice, France, pages:85-102,1998.

[34] Dugan J.B., Trivedi K., Geist R., Nicola V., Extended Stochastic Petri-Nets: Application and Analysis, Proceedings of the Tenth International Symposium on Computer Performance Modeling, Measurement and Evaluation, pages: 507-519, 1984.

[35] Dohi T., Goseva-Popstojanova K., Trivedi K.S., Estimating Software Rejuvenation Schedule in High Assurance Systems, Computer Journal, 44, pages 473-485, 2001

[36] Dohi T., Goseva-Popstojanova K., Trivedi K.S., Statistical Non-Parametric Algorithms to Estimate the Optimal Software Rejuvenation Schedule. Proceedings of the 2000 Pacific Rim International Symposium on Dependable Computing, pages 77-84, IEEE Computer Society Press, Los Alamitos CA. 2000

[37] Dohi T., Iwamoto K., Okamura H., Kaio N., Discrete Availability Models to Rejuvenate a Telecommunication Billing Application, IEICE Transactions on Communications, pages 2931-2939, Volume E86-B (10), 2003.

[38] Fricks R., Puliafito A., Telek M., Trivedi K., Applications of Non-Markovian Stochastic Petri Nets, ACM SIGMETRICS Performance Evaluation Review, Vol. 26, Issue 2 pages: 15 - 27, 1998.

[39] Garg S., Puliafito A., Telek M., Trivedi K.S., Analysis of Software Rejuvenation Using Markov Regenerative Stochastic Petri Nets. Proceedings of the 6th International Symposium on Software Reliability Engineering, pages 24-27, IEEE Computer Society Press, Los Altimos, CA, 1995.

[40] German R., Telek M., Formal Relation of Markov Renewal Theory and Supplementary Variables in the Analysis of Stochastic Petri Nets, Proceedings of the 8th International Workshop on Petri Nets and Performance Models, page: 64, 1999.

[41] German R., Performance Analysis of Communication Systems, Modeling with non-Markovian Stochastic Petri Nets, John Wiley and sons, 2000.

[42] German R., New Results for the Analysis of Deterministic and Stochastic Petri Nets, Proceedings of the International Computer Performance and Dependability Symposium on Computer Performance and Dependability Symposium, page114, 1995

[43] German R., Markov regenerative Stochastic Petri Nets with General Execution Policies: Supplementary Variable Analysis and a Prototype Tool, Performance Evaluation, Vol. 39, Issue 1-4, pages: 165-188, 2000.

[44] German R., Lindemann C.: Analysis of Stochastic Petri Nets by the Method of Supplementary Variables, Performance Evaluation, v.20 No.1-3, pages: 317-335, May 1994.

[45] German R., Logothetis D., Trivedi K., Transient Analysis of Markov Regenerative Stochastic Petri Nets: A Comparison of Approaches, Proceedings of the Sixth International Workshop on Petri Nets and Performance Models, page 103, 1995.

[46] Gorrieri R., Rocetti M., Towards Performance Evaluation in Process Algebra**,** Proceedings of the Third International Conference on Methodology and Software Technology: Algebraic Methodology and Software, Springer Verlag London UK, pages: 289-296, 1993

[47] Goseva-Popstojanova K., Trivedi K., Stochastic Modeling Formalisms for Dependability, Performance, and Performability, Lecture Notes In Computer Science; Vol. 1769, Performance Evaluation: Origins and Directions, pages: 403-422, 2000.

[48] Gross D., Harris C.M., Fundamentals of Queuing Theory, John Wiley & Sons, 1985.

[49] Henderson W., Lucic D., Applications of Generalized Semi-Markov Processes to Stochastic Petri Nets, Proceeding of IFIP TC7/WG 7.3 International Seminar on Performance of Distributed and Parallel Systems, Kyoto, Japan, 1988.

[50] Hermanns H., Herzog U., Katoen J-P., Process Algebra for Performance Evaluation, Theoretical Computer Science, vol.274, issue1-2, pages: 43--87, 2002.

[51] Herzog U., A concept for Graph Based Stochastic Algebras, generally Distributed Activity Times, and Hierarchical Modeling, Proceedings of Process Algebra and Performance Modeling PAPM, pages. 1-20, Torino, Italy, 1996.

[52] Herzog U., Formal Description, time and Performance Analysis: A Framework, Technical Report 15/90, IMMD VII, Friedrich-Alexander-Universitat, Erlangen-Nurnberg, Germany, September 1990.

[53] Hillston J., Modeling and Simulation Course, www.dsc.ed.ac.uk/teaching/cs4, 2003 consulted in June 2004.

[54] Hillston J., A compositional Approach to Performance Modeling, Distinguished Dissertation in Computer Science. Cambridge University Press, New York, NY, USA 1996.

[55] Horvath A., Puliafito A., Scarpa M., Telek M., Analysis and Evaluation of non-Markovian Stochastic Petri-Nets, Computer Performance Evaluation. International conference N^{o}11, Schaumburg IL , U.S., Lecture Notes in Computer Science, vol.1786, pages:171-187, 2000.

[56] Howard R., Dynamic Probabilistic Systems: Semi-Markov and Decision Systems Volume II, Wiley 1971.

[57] Huang Y., Kintala C., Kolettis N., Fulton N.D., Software Rejuvenation: Analysis, Module and Application. Proceedings of the 25th International Symposium on Fault Tolerant Computing, pages 381-390, 1995. IEEE CS Press.

[58] Janssen J., Manca R., Numerical Solution of non-Homogeneous Semi-Markov Processes in Transient Case. Journal of Methodology and Computing in Applied Probability, pages 271-293, Volume 3 (3), September, 2001.

[59] Janssen J., Manca R., Applied Semi-Markov Processes. Springer US, 2006.

[60] Jensen K., Rosenberg G., (eds.), High-level Petri Nets. Theory and Application. ISBN: 3-540-54125 X, Springer-Verlag, 1991.

[61] Katoen J.P., D'Argenio P.R., General Distributions in Process Algebra, Lectures on Formal Methods and Performance Analysis. Lecture notes in Computer Science 2090. Springer Verlag, Berlin, pages: 375-429, 2001.

[62] Kulkarni, V.G., Modeling and Analysis of Stochastic Systems, Chapman & Hall, 1995.

[63] Lindemann C., Performance Modeling with Deterministic and Stochastic Petri Nets, John Wiley & Sons, 1998.

[64] Lindemann C., An Improved Numerical Algorithm for Calculating Steady State Solutions of Deterministic and Stochastic Petri-Net Models. International workshop on Petri nets and performance models N^{o}4, Melbourne, Australia, vol.18, issue 1, pages:79-95, 1993.

[65] Lindemann C, Thummler A., Numerical Analysis of Generalized Semi-Markov Processes, Technical Report: Department of Computer Science, University of Dortmund, 2003.

[66] Lu D., Fast Search Method for Enabling a Computer to Find Elementary Loops in a Graph. http://www.patentstorm.us/patents/6438734-description.html, August 20, 2002.

[67] Matthes K., Zur Theorie der Bedienungsprozesse, in Transactions of the 3rd Prague Conference on Information Theory, 1962.

[68] Mehdi J., Stochastic Processes, John Wiley and Sons Inc.1994.

[69] Meyer J.F., On Evaluating the Performability of Degradable Computing Systems. IEEE Transactions on Computers, Pages 720-731, Vol. 29 (8) August 1980.

[70] Neuts M., Probability Distributions of Phase Type, Technical Report, Department of Mathematics, University of Louvain, pages: 173-206,1975.

[71] Petriu D., Woodside M., Analyzing Software Requirements Specifications for Performance, Performance, Proceedings of the 3rd International Workshop on Software and Performance, Rome, pages 1-9, 2002.

[72] Pressman R. S., Software Engineering: A Practitioner's Approach, Prentice-Hall, 1996.

[73] Pyke R., Markov Renewal Processes with Finitely Many States, The Annals of Mathematical Statistics, vol. 32, No. 4, pages: 1243-1259, 1961.

[74] Rinsaka K., Dohi T., A Faster Estimation Algorithm for Periodic Preventative Rejuvenation Schedule Maximizing System Availability. Lecture Notes in Computer Science, vol. 4526, pages 94-109, Springer Berlin, 2007

[75] Schassberger R., Insensitivity of Steady State Distributions of Generalized Semi-Markov Processes, The Annals of Probability, vol. 6, No. 1, pages: 85-93, 1978.

[76] Smith U. C., Performance Engineering of Software Systems, Addison-Wesley, 2002.

[77] Smith C., Woodside M., Performance Validation at Early stages of Software Development, The Journal of Systems and Software, vol. 49, issue 1, pages: 63-80,1999.

[78] Stewart W.J., Introduction to the Numerical Solution of Markov Chains, Princeton University Press, ISBN 0691036993, 1994.

[79] Strulo B., Process Algebra for Discrete Event Simulation, PhD. Thesis, Department of Computing, Imperial College of Science, Technology and Medicine. University of London, 1993

[80] Suzuki H., Dohi T., Kaio N., Trivedi K., Maximizing Interval Reliability in Operational Software System with Rejuvenation, Proceedings of the 14th International Symposium on Software Reliability Engineering, pages 479-490, vol. 17 (20), 2003.

[81] Szwarcfiter J.L., Lauer P.E., A Search Strategy for the Elementary Cycles of a Directed Graph. BIT Numerical Mathematics, Vol. 16 (2), June, 1976.

[82] Trivedi K., Reliability and Performability Techniques and Tools, A survey, Messung, Modellierung und Bewertung von Rechen- und Kommunikations systemen, vol. 7., pages: 21-23, 1993

[83] Waters G., Linington P., Akehurst D., Symes A., Communication Software Performance Prediction, 13th UK Workshop on Performance Engineering of Computer and Telecommunication Systems, pages: 381-389, Ilkley, West Yorkshire, 1997.

[84] Winograd S., Coppersmith D., Matrix multiplication via arithmetic progressions. Proceedings of the 19th annual ACM Symposium on Theory of Computing, New York, United States, pages: 1-6, 1987.

[85] Woodside M., Software Performance Evaluation by Models, Lecture Notes in Computer Science; Vol. 1769, Performance Evaluation: Origins and Directions, pages: 283 - 304, 2000.