University of Twente
The Netherlands

Model Checking Markov Reward Models with Impulse Rewards

Maneesh Khattri
Mhd. Reza M. I. Pulungan

A thesis submitted to the department of Computer Science
of the University of Twente for the partial fulfillment
of the requirements of the degree of Master of Science in Telematics

Graduation Committee:
Dr. ir. Joost-Pieter Katoen (mtd.)
Dipl. Inform. Lucia Cloth

Enschede, the Netherlands
June 2004

Abstract

A Markov Reward Model (MRM) is a Stochastic Process satisfying the Markov property. MRMs have been used for the simultaneous analysis of performance and dependability issues of computer systems, sometimes referred to as performability.

Previous work for Model Checking MRMs [Bai00, Bai02, Hav02] is restricted to state-based reward rates only. However, greater insight into the operation of systems can be obtained when impulse rewards are incorporated in these models which express the instantaneous cost imposed with the change of state of systems.

The main aim of this thesis is to enable Model Checking of MRMs with impulse rewards. In this thesis methods for Model Checking MRMs have been extended for the incorporation of impulse rewards. To that end, the syntax and semantics of the temporal logic CSRL (Continuous Stochastic Reward Logic) have been adapted and algorithms have been developed and implemented that allow for the automated verification of properties expressed in that logic.

Acknowledgements

We thank our supervisors Dr. ir. Joost-Pieter Katoen and Dipl. Inform. Lucia Cloth for their support and guidance during the course of this project work. We especially extend our gratitude to Dr. ir. Joost-Pieter Katoen for directing us towards a project, for patiently reading versions of this report and helping us in improving it.

We also thank Dr. ir. Henk Alblas, Drs. Jan Schut and Ms. Belinda Jaarsma Knol for helping us on many occasions during our stay in the Netherlands. We're indebted to family members and friends for their encouragement and help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the motivation, objective and structure of this thesis. A brief introduction to Markov Reward Models *is presented in section* 1.1. *Section* 1.2 *gives a short overview of* Model Checking. *The motivation behind this thesis is presented in section* 1.3. *In section* 1.4*, the objective of this project is stated and section* 1.5 *describes the structure of this thesis.*

1.1 Markov Reward Model

Many events occur in a disorderly or random fashion. For instance the arrival of a customer to an airline counter or the time it will take to serve the customer. In mathematics such events are described by random functions for which if the value of the function is given at a certain point of time, its future value can only be determined with a certain probability and not precisely. Such functions to represent random events are referred to as *Stochastic Processes* [How71, Hav98].

One special class of Stochastic Processes is referred to as the *Markov Process*. A Markov Process is essentially a Stochastic Process which satisfies the *Markov Property*. The Markov Property states that "given the present state, the knowledge of the past states does not influence the future state." This is sometimes referred to as the memoryless property. A *Markov Chain* is a Markov Process whose state space is finite or countably infinite.

A *Markov Reward Model* (MRM) is such a Markov Chain. It is augmented with a *reward assignment function*. The reward assignment function assigns a *state-based reward rate* to each state such that the residence time in a state entails the accumulation of overall reward gained based on the state-based reward rate. The reward assignment function also assigns an *impulse reward* to each transition such that the occurrence of the transition results in a gain in the overall reward governed by the impulse reward assigned to the transition.

The application of Markov Reward Models to computer systems can be traced back to the work of Professor John F. Meyer of the University of Michigan, Ann Arbor [Mey80] which concerns the simultaneous analysis of *performance* and *dependability* of computer systems. Whilst performance in computer systems is the efficacy by which the system delivers services under the assumption that the services delivered conform to the specification, dependability is the ability of the system to deliver services that conform to the specification. Although these two issues may be analyzed independently, simultaneous analysis becomes imperative when the performance of the system *degrades* under the presence of behavior that does not conform to the specification. This analysis is sometimes referred to as *performability* [Mey95].

The definition of the *performability measure* is based on a *performability variable* (Y). The performability variable is realized over a *base stochastic model* and incorporates an *utilization interval* and an *accomplishment set* in which the performability variable takes its values. An example of such an accomplishment set is the set of all non-negative real numbers $\mathbb{R}_{\geq 0}$. The performability measure for a subset of the accomplishment set is then the probability that the performability variable interpreted over the stochastic process takes values in the subset. For instance, a subset of the accomplishment set $\mathbb{R}_{\geq 0}$ is $[0, y]$, $y \in \mathbb{R}_{\geq 0}$.

The accumulated reward over an MRM in a finite interval can be considered to be a performability variable. Then performability over an MRM for a finite interval $[0, t]$ and for a set $[0, y]$, $y \in \mathbb{R}_{\geq 0}$ is defined as the probability that the accumulated reward is in $[0, y]$ over a utilization interval $[0, t]$.

1.2 Model Checking

Model Checking [Cla99, Kat02] is a *formal verification strategy.* The verification of systems requires first a precise *specification* of the system. A *model* of the system is then constructed whose behavior should correspond to the specification. Once such a model is available, *statements* about functional correctness of the system can be made. The specification of these statements is made in mathematical logic such as *temporal logic* interpreted over the model. Subsequently, a set of statements about the functional correctness of the system are made.

Once the model of the system and the statements about the functional correctness of the system are available, it has to be ascertained whether the model satisfies these statements. This is either achieved by simulation tests or by formal verification methods. If the model satisfies all these statements then it is said to be *correct* and this strategy to ascertain correctness is called Model Checking. The specification of a model of a system and statements about its correctness in mathematical terms requires careful insight into the modeling concepts and the functioning of the system to ensure that the model portrays the system precisely.

The relationship between the system and the model in most cases however cannot be clearly established. One primary cause of this is that models are usually simplified and designed to capture some essence but not all aspects of systems. Yet even such simplified models cannot be made manually. Frequently, these models are specified at an abstract level which can automatically be refined into models at detailed specification levels. Formal verification provides tools to capture such modeling concepts.

1.3 Motivation

In recent times as performability requirements on computer systems become more acute, models based on MRM are gaining more importance and so do verification techniques for these models. This is the inspiration to investigate model checking applied to the field of performability. Many qualitative and quantitative statements can be made about systems modeled in terms of such MRM by means of Model Checking. Previous work for Model Checking MRMs [Bai00, Bai02, Hav02] is restricted to state-based reward rates only. However, greater insight into the operation of systems can be obtained when impulse rewards are incorporated in these models which express the instantaneous cost imposed with the change of state of systems.

The occurrence of this instantaneous cost in terms of energy consumption can be observed in modern-day cellular phones. The phone periodically moves to an *idle* state. It then stays in such a state continuously unless a call is initiated or received or if a location hand-over becomes necessary. If a call is received then the phone has to transition to a state where it can handle the call. During this transition instantaneous costs related to tasks such as preparing to ring the ringer are endured. This behavior of accumulating instantaneous costs can be modeled by impulse reward functions.

This ability to model instantaneous costs is the motivation to extend Model Checking procedures defined for MRMs with only state-based reward rates to *include impulse reward functions*. A further motivation is to develop efficient and numerically stable algorithms for model checking MRMs.

1.4 Objective

To develop modeling formalisms and practical algorithms for model checking Markov Reward Models with state-based reward rate *and* impulse rewards. To fulfill this objective the following tasks are distinguished:

1. Background study of Markov Processes, Markov Reward Models and Performability measures.

2. Extension of definitions, model and logic to incorporate impulse rewards.

3. Survey of numerical methods to compute measures defined over MRMs.

4. Development and implementation of algorithms for Model Checking MRMs.

5. Development of an example application which is modeled as an MRM and performing experiments with the example.

6. Comparison of the performance of new algorithms with existing methods for models with only state-based reward rates.

1.5 Structure

This thesis is organized as follows:

Chapter 2: *Markov Processes* describes background theory of Markov Processes. It presents a brief introduction to Stochastic Processes, Markov Processes and Discrete & Continuous-Time Markov Chains.

Chapter 3: *Markov Reward Model (MRM)* describes Markov Reward Models. It presents the formal definition of MRMs and measures interpreted over these MRMs. Subsequently a logic (CSRL) to specify properties over MRMs is presented.

Chapter 4: *Model Checking MRMs* describes the model checking procedure to check the validity of properties specified in CSRL interpreted over MRMs. Numerical methods which have been developed to check the validity of CSRL properties are presented. This constitutes the main part of this thesis.

Chapter 5: *Experimental Results* describes the implementation of model checking procedures and several experiments using this implementation. It also presents a comparison of the performance of the numerical methods with existing methods for models with only state-based reward rates.

Chapter 6: *Conclusion* presents conclusions drawn in the context of this thesis. This chapter also presents indications towards future work in Model Checking MRMs with state-based reward rates and impulse rewards.

Chapter 2

Markov Processes

This chapter presents background theory of Markov Processes. A brief introduction to Stochastic Processes *is presented in section* 2.1. *Section* 2.2 *gives an overview of* Markov Processes*. Discussion about Discrete and Continuous-Time Markov Chains is presented in section* 2.3 *and in section* 2.4 *respectively. Section* 2.5 *discusses Continuous-Time Markov Chains augmented with a labeling function.*

2.1 Stochastic Processes

A Stochastic Process is a collection of random variables $\{X(t) | t \in \mathcal{T}\}$ indexed by a parameter t which can take values in a set $\mathcal{T}$ which is the time domain. The values that $X(t)$ assumes are called *states* and the set of all possible states is called the *state space* $\mathcal{I}$. Both set $\mathcal{I}$ and $\mathcal{T}$ can be discrete or continuous, leading to the following classification:

1. *Discrete-state discrete-time stochastic processes* can be used for instance to model the number of patients that visit a physician where the number of patients represents states and time is measured in days.

2. *Discrete-state continuous-time stochastic processes* can be used for instance to model the number of people queueing in an airplane ticketing counter where the number of people in the queue represents states.

3. *Continuous-state discrete-time stochastic processes* can be used for instance to model the volume of water entering a dam where the volume of water entering the dam represents states and time is measured in days.

4. *Continuous-state continuous-time stochastic processes* can be used for instance to model the temperature of a boiler where the temperature of the boiler represents states.

At a particular time $t' \in \mathcal{T}$, the random variable $X(t')$ may take different values. The distribution function of the random variable $X(t')$ for a particular

$t' \in \mathcal{T}$ is defined as:

$$F(x', t') = \Pr\{X(t') \leq x'\},$$

which is also called the cumulative density function of the random variable or the first-order distribution of the stochastic process $\{X(t) | t \in \mathcal{T}\}$. This function can be extended to the n-th joint distribution of the stochastic process as follows:

$$F(\underline{x}', \underline{t}') = \Pr\{X(\underline{t}'_1) \leq \underline{x}'_1, \ldots, X(\underline{t}'_n) \leq \underline{x}'_n\},$$

where $\underline{x}'$ and $\underline{t}'$ are vectors of size n, $\underline{x}'_i \in \mathcal{I}$ and $\underline{t}'_i \in \mathcal{T}$ for all $1 \leq i \leq n$.

An *independent process* is a stochastic process where the state being occupied at a certain time does not depend on the state(s) being occupied in any other time-instant. Mathematically, an *independent process* is a stochastic process whose n-th order joint distribution satisfies:

$$F(\underline{x}', \underline{t}') = \prod_{i=1}^{n} F(\underline{x}'_i, \underline{t}'_i) = \prod_{i=1}^{n} \Pr\{X(\underline{t}'_i) \leq \underline{x}'_i\}.$$

A stochastic process can also be a *dependent process* in which case some form of dependence exists among successive states.

2.2 Markov Processes

One form of a dependent process in which there is a dependence only between two successive states is called a *Markov process*. Such dependence is called *Markov dependence* or *first-order dependence*. A stochastic process $\{X(t) | t \in \mathcal{T}\}$ is a Markov process if for any $t_0 < t_1 < \cdots < t_n < t_{n+1}$, the distribution of $X(t_{n+1})$, given $X(t_0), \cdots, X(t_n)$, only depends on $X(t_n)$, or mathematically:

$$\Pr\{X(t_{n+1}) \leq x_{n+1} | X(t_0) = x_0, \ldots, X(t_n) = x_n\} = \Pr\{X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n\},$$

which is referred to as the *Markov property*. This is to say that the immediate future state in a Markov process depends only on the state being occupied currently.

A Markov process is called *time-homogeneous* if it is invariant to time shifts which means that the behavior of the process is independent of the time of observation. For any $t_1 < t_2$, x_1 and x_2:

$$\Pr\{X(t_2) \leq x_2 | X(t_1) = x_1\} = \Pr\{X(t_2 - t_1) \leq x_2 | X(0) = x_1\}.$$

If the state space $\mathcal{I}$ of a Markov process is discrete, the Markov process is called a *Markov chain*. Hence two types of Markov chains can be identified, namely *Discrete-Time Markov Chains* and *Continuous-Time Markov Chains*.

2.3 Discrete-Time Markov Chains

A Discrete-Time Markov Chain (DTMC) is a stochastic process such that the state space $\mathcal{I}$ and the set $\mathcal{T}$ are both discrete, and the stochastic process satisfies the Markov property, which in the discrete-state and discrete-time case is defined as:

$$\Pr\{X_{n+1}) = i_{n+1} | X_0 = i_0, \cdots, X_n = i_n\} = \Pr\{X_{n+1} = i_{n+1} | X_n = i_n\},$$

where $\mathcal{T} = \{0, 1, 2, \ldots\}$ and $i_0, \ldots i_{n+1} \in \mathcal{I}$.

Let $p_i(n)$ denote the probability of being in state i at time n, and the conditional probability $p_{j,k}(m, n) = \Pr\{X_n = k | X_m = j\}$ denote the probability of being in state k at time n, given that at time m the DTMC is in state j. Since in time-homogeneous Markov chains, the transition probabilities only depend on the time difference, this conditional probability can be written as $p_{j,k}(l) = \Pr\{X_{m+l} = k | X_m = j\}$, which is called the l-step transition probability. Hence, the l-step transition probability $p_{j,k}(l)$ denotes the probability of being in state k after l steps given that the current state is j. The 1-step probabilities are $p_{j,k}(1)$ or simply $p_{j,k}$ and the 0-step probabilities are the initial distribution of the DTMC. A DTMC is described by the initial probabilities $\underline{p}(0)$ and the 1-step probabilities, which is represented by the state-transition probability matrix $\boldsymbol{P}$.

Example 2.1 *A DTMC can be conveniently represented graphically as a labeled directed graph. The vertices in the graph represent states in the DTMC and the name of the state is in the vertex that represents the state. A transition is represented by an edge in the graph. The probability of a transition is placed near the edge representing the transition in question. Figure 2.1 is an example of such a graph.*

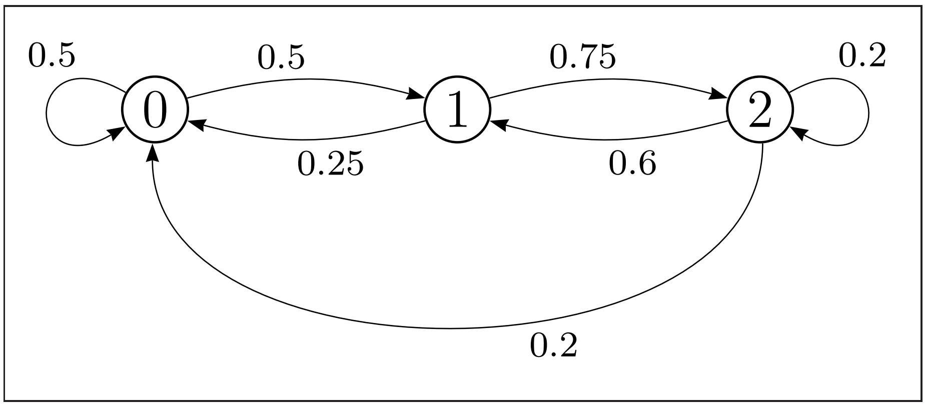


Figure 2.1: A DTMC Represented in Labeled Directed Graph

Figure 2.1 depicts a DTMC with three states, named 0, 1 *and* 2*. There are seven transitions in the DTMC. These transitions are shown together with their* *probability, for instance, transition from state* 0 *to state* 1 *occurs with probability* 0.5*. From the figure the* 1*-step probabilities are given by:*

$$\boldsymbol{P} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0 & 0.75 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}.$$

The Markov property can only be satisfied if the state residence time in the DTMC has a memoryless discrete distribution. Since the geometric distribution is the only memoryless discrete distribution, it follows that the state residence time in a DTMC is geometrically distributed. Hence, for every state $i \in \mathcal{I}$ in the DTMC a non-negative real value $\boldsymbol{P}_{i,i} = 1 - \sum_{i \neq j} \boldsymbol{P}_{i,j}$, is associated such that the residence time distribution in state i (the probability to reside in state i for n steps) is:

$$F_i(n) = (1 - \boldsymbol{P}_{i,i}) \cdot \boldsymbol{P}_{i,i}^{n-1}, n \geq 0.$$

2.3.1 Transient Analysis

Transient analysis aims at determining the probability with which the DTMC occupies a state after a given number of observation steps have occurred. This probability is referred to as the *state occupation probability*. These state occupation probabilities are given by:

$$\underline{p}(n) = \underline{p}(0) \cdot \boldsymbol{P}^n,$$

where $\underline{p}(0)$ is the initial distribution of the DTMC and $\boldsymbol{P}$ is the state-transition probability matrix. The state occupation probabilities, which is contained in $\underline{p}(n)$ express the transient behavior of the DTMC.

Example 2.2 *Using the DTMC presented in figure 2.1,and* $p_0(0) = 1$*;* $p_i(0) = 0$ *for* $i = 1, 2$*; the state occupation probabilities after* 3 *steps are as follows:*

$$\underline{p}(3) = \underline{p}(0) \cdot \boldsymbol{P}^3 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0 & 0.75 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}^3 = \begin{bmatrix} 0.325 & 0.412\,5 & 0.262\,5 \end{bmatrix}.$$

These state occupation probabilities describe the probabilities of ending in states after 3 *steps starting from state* 0*. For instance, after* 3 *steps, starting from state* 0*, the DTMC will be in state* 2 *with probability* 0.2625*. State occupation probabilities after more number of steps have elapsed can also be obtained. For instance the state occupation probabilities after* 15 *and* 25 *steps are as follows:*

$$\begin{aligned} \underline{p}(15) &= \begin{bmatrix} 0.311\,1 & 0.355\,67 & 0.333\,23 \end{bmatrix}, \\ \underline{p}(25) &= \begin{bmatrix} 0.311\,11 & 0.355\,56 & 0.333\,33 \end{bmatrix}. \end{aligned}$$

In the previous example it can be observed that after a certain number of steps, the state occupation probabilities converge. It would be interesting to know if the converged probabilities can be determined directly since for some measures these converged probabilities will suffice to describe the behavior of the DTMC. However, such converged probabilities do not exist for all DTMC. The conditions under which these converged probabilities exist are presented in [Hav98].

2.3.2 Steady-State Analysis

The steady-state analysis aims at determining the state occupation probabilities after an unbounded number of steps have occurred in a DTMC: $v_i = \lim_{n\to\infty} p_i(n)$, which are called the steady-state probabilities. If this limit exists, these steady-state probabilities are described by a system of linear equations:

$$\underline{v} = \underline{v} \cdot \boldsymbol{P}, \textstyle\sum_i v_i = 1, 0 \leq v_i \leq 1.$$

Vector $\underline{v}$ is referred to as the steady-state probability vector of the DTMC, which describes the steady-state distribution of the DTMC.

Example 2.3 *Using the DTMC presented in figure 2.1, in a long run, after infinite steps occur in the DTMC, the state occupation probabilities are given by the steady-state probabilities. For the DTMC, the steady-state probabilities are:*

$$\underline{v} = \underline{v} \cdot \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0 & 0.75 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}, \textstyle\sum_i v_i = 1, 0 \leq v_i \leq 1 \Rightarrow \underline{v} = \begin{bmatrix} \frac{14}{45} & \frac{16}{45} & \frac{1}{3} \end{bmatrix}.$$

These steady-state probabilities can be interpreted as the probabilities of discovering that the DTMC is in some state after it has been running for a long time. They can also be interpreted as the fraction of time the DTMC stays in some state in the long run. Thus for the example, it can be said that after a long running time the DTMC will be in state 2 *with probability* $\frac{1}{3}$ *or the fraction of running time that the DTMC spends in state* 2 *is* $\frac{1}{3}$.

2.4 Continuous-Time Markov Chains

A Continuous-Time Markov Chain (CTMC) is a stochastic process such that the state space $\mathcal{I}$ is discrete, the set $\mathcal{T}$ is continuous, and the stochastic process satisfies the Markov property namely:

$$\Pr\{X(t_{n+1}) = x_{n+1} | X(t_0) = x_0, \cdots, X(t_n) = x_n\} = \Pr\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n\}.$$

The Markov property can only be satisfied if the state residence time in the CTMC has a memoryless continuous distribution. Since the *negative exponential distribution* is the only memoryless continuous distribution, it follows that the state residence time in a CTMC is negative exponentially distributed. Hence, for every state $i \in \mathcal{I}$ in the CTMC a random variable with parameter a non-negative real value μ_i, referred to as the *rate*, is associated such that the residence time distribution in state i is:

$$F_i(t) = 1 - e^{-\mu_i \cdot t}, t \geq 0.$$

Beside the state residence time distribution with rate μ_i, several delays depending on the number of transitions are associated with every state with rate $\boldsymbol{R}_{i,j}$. The total rate of taking an outgoing transition from state i is $E(i) = \sum_j \boldsymbol{R}_{i,j}$. Note that the rate of residence in state i is now $\mu_i = E(i)$. These delays can conveniently be represented in a rate matrix $\boldsymbol{R}$.

The operation of CTMCs can be imagined as follows, at any given instant the CTMC is said to be in one of the states. Let the CTMC enter state i at some observation instant. Subsequently, it will make a transition to one of the states j after residing in state i for a negative exponentially distributed interval of time given by the following distribution:

$$F_{i,j}(t) = 1 - e^{-\boldsymbol{R}_{i,j} \cdot t}.$$

Since the delays assigned to various transitions from state i can be different, the transition corresponding to the fastest rate will take the shortest time to take place. Hence at the observation instant all transitions from state i are said to be in a *race condition*. Due to this property the probability that a certain transition $i \rightarrow j$ is successful in relation to other transitions from state i is:

$$P(i,j) = \frac{\boldsymbol{R}_{i,j}}{\sum_k \boldsymbol{R}_{i,k}}.$$

Consequently, the probability of moving from state i to a state j within time t is given by $P(i,j) \cdot (1 - e^{-E(i) \cdot t})$. For most measures defined over CTMCs the specification of a CTMC involves the specification of the initial probability vector $\underline{p}(0)$ and the rate matrix $\boldsymbol{R}$ only.

2.4.1 Transient Analysis

For many measures that are defined over CTMCs it is interesting to know the probability with which the CTMC occupies a state after a given observation interval has elapsed. This probability is referred to as the *state occupation probability*. The analysis used to determine the state occupation probabilities in this manner is called *transient analysis* of CTMCs. These state occupation probabilities are described by a linear system of differential equations:

$$\underline{p}'(t) = \underline{p}(t) \cdot \boldsymbol{Q}, \tag{2.1}$$

where

$$\boldsymbol{Q} = \boldsymbol{R} - Diag(E),$$

is the *infinitesimal generator matrix*.

In many practical situations, *uniformization* is used to perform transient analysis of CTMC by analyzing the uniformized Discrete Time Markov Chain (DTMC). To generate the uniformized process of the CTMC, specified by initial probability distribution $\underline{p}(0)$ and rate matrix $\boldsymbol{R}$, first obtain the infinitesimal generator matrix $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{1,1} & \boldsymbol{Q}_{1,2} & \cdots & \boldsymbol{Q}_{1,n} \\ \boldsymbol{Q}_{2,1} & \boldsymbol{Q}_{2,2} & \cdots & \boldsymbol{Q}_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{Q}_{n,1} & \boldsymbol{Q}_{n,2} & \cdots & \boldsymbol{Q}_{n,n} \end{bmatrix} \text{ where } \boldsymbol{Q}_{i,i} = -\sum_{j=1}^{n} \boldsymbol{Q}_{i,j},$$

a rate of residence Λ is assigned to each state where $\Lambda \geq \max_i(-\boldsymbol{Q}_{i,i})$. The 1-step probability matrix for the uniformized process is found in the following fashion:

$$\boldsymbol{P} = \boldsymbol{I} + \frac{\boldsymbol{Q}}{\Lambda} \text{ where } \boldsymbol{I} \text{ is an Identity matrix of cardinality } (n \times n).$$

Once such a uniformized process is available, let $\{N_t : t \geq 0\}$ be a Poisson process with rate Λ, $\underline{p}(t)$, the distribution of state occupation probability at time t, is:

$$\underline{p}(t) = \sum_{i=0}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^i}{i!} \cdot \underline{p}(0) \cdot \boldsymbol{P}^i. \tag{2.2}$$

2.4.2 Steady-State Analysis

Consider a CTMC specified by an initial probability distribution $\underline{p}(0)$ and a rate matrix $\boldsymbol{R}$. For many measures defined over such a CTMC it suffices to consider the *steady-state distribution*. The steady-state analysis aims at determining the state occupation probabilities after an unbounded observation interval is allowed to elapse: $p_i = \lim_{t\to\infty} p_i(t)$. For finite CTMC this limit always exists; hence from equation (2.1) $\underline{p}'(t) = \underline{0}$. Consequently, these steady-state probabilities are described by a system of linear equations:

$$\underline{p} \cdot \boldsymbol{Q} = \underline{0}, \sum_{i \in \mathcal{I}} p_i = 1. \tag{2.3}$$

When the steady-state distribution depends on the initial-probability distribution then a graph analysis to determine the bottom-strongly connected components can be performed. This is elaborated upon in chapter 3.

2.5 Labeled Continuous-Time Markov Chain

Continuous-Time Markov Chains can be further augmented with a labeling function. This labeling functions assigns a set of statements to each state which represent certain qualitative properties of the state in question. This enables qualitative statements about the system modeled as a CTMC to be made.

2.5.1 Atomic Propositions

Atomic propositions are the most elementary statements that can be made, cannot be further decomposed and can be justified to be `true` or `false`. For instance, "*It is Sunday*" is an atomic proposition. The finite set of all atomic propositions is referred to as AP. The selection of such a set AP determines the qualitative aspects that can be expressed about a system.

2.5.2 Interpretation Function: Labeling (Label)

$Label : S \longrightarrow 2^{AP}$: A labeling function *Label* assigns to each state s a set of atomic propositions $Label(s) \in AP$ that are valid (`true`) in state s. The function *Label* indicates which atomic propositions are valid in which states. A state s for which the atomic proposition $p \in AP$ is valid i.e. $p \in Label(s)$ is called a p-state.

Definition 2.1 *A Continuous-Time Markov Chain (CTMC) $\mathcal{C}$ is a three-tuple $(S, \boldsymbol{R}, Label)$ where S is a finite set of states, $\boldsymbol{R} : S \times S \longrightarrow \mathbb{R}_{\geq 0}$ is a function. $\boldsymbol{R}$ is called the* rate matrix *of the CTMC, where $\boldsymbol{R}_{s,s'}$ is the rate of moving from state s to s'. It is said that there is a transition from state s to state s' if and only if $\boldsymbol{R}_{s,s'} > 0$. $Label : S \longrightarrow 2^{AP}$ is a labeling function.*

The total rate of taking an outgoing transition from state $s \in S$ is given by $E(s) = \sum_{s' \in S} \boldsymbol{R}_{s,s'}$. The probability of moving from state s to any state within time t is $(1 - e^{-E(s)t})$. The probability of moving from state s to a state s' is:

$$P(s, s') = \frac{\boldsymbol{R}_{s,s'}}{E(s)}.$$

The probability of making a transition from state s to s' within time t is $\frac{\boldsymbol{R}_{s,s'}}{E(s)} \cdot (1 - e^{-E(s) \cdot t})$. Note that this definition of a CTMC is different from the original definition of a CTMC in the sense that self-transitions are allowed to occur. This implies that after a negative exponentially distributed residence time in a state has elapsed the CTMC may transition to the same state.

Example 2.4 *Consider a WaveLAN modem which is designed to be energy- efficient. Typical operating modes in such an interface are* off, sleep, idle, receive *and* transmit *[Pau01]. In the transmit mode it is transmitting and is receiving data in the receive mode. In the idle mode it is idle but ready to either transmit or to receive. In the sleep mode the interface is powered down. Consequently, it is neither ready to receive nor to transmit but listens for some incoming transmissions. When in sleep mode the interface is powered up after some time has elapsed and subsequently is ready either to transmit or to receive. Such a WaveLAN modem is modeled as a labeled CTMC in figure 2.2.*

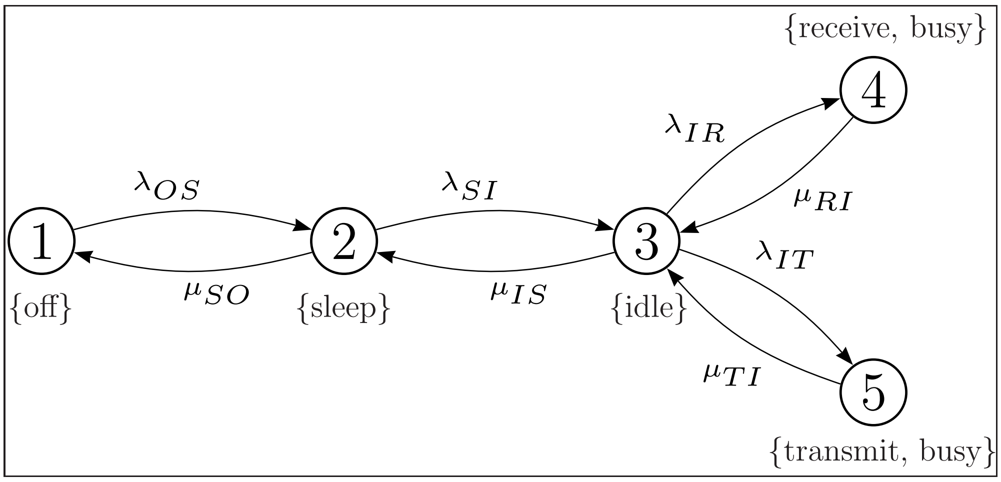


Figure 2.2: WaveLAN Modem Modeled as a Labeled CTMC

This CTMC can be specified as $\mathcal{C} = (S, \boldsymbol{R}, Label)$ *where* $S = \{1, 2, 3, 4, 5\}$,

$$\boldsymbol{R} = \begin{bmatrix} 0 & \lambda_{OS} & 0 & 0 & 0 \\ \mu_{SO} & 0 & \lambda_{SI} & 0 & 0 \\ 0 & \mu_{IS} & 0 & \lambda_{IR} & \lambda_{IT} \\ 0 & 0 & \mu_{RI} & 0 & 0 \\ 0 & 0 & \mu_{TI} & 0 & 0 \end{bmatrix} ; \textit{ and } \begin{array}{l} Label(1) = \{\text{off}\}, \\ Label(2) = \{\text{sleep}\}, \\ Label(3) = \{\text{idle}\}, \\ Label(4) = \{\text{receive,busy}\}, \\ Label(5) = \{\text{transmit,busy}\}. \end{array}$$

From the rate matrix, the total rate of taking an outgoing transition for each state $s \in S$ *(*$E(s)$*) can be calculated:*

$$\begin{aligned} E(1) &= \lambda_{OS}, \\ E(2) &= \lambda_{SI} + \mu_{SO}, \\ E(3) &= \lambda_{IR} + \lambda_{IT} + \mu_{IS}, \\ E(4) &= \mu_{RI}, \\ E(5) &= \mu_{TI}. \end{aligned}$$

In figure 2.2 by the application of the labeling function certain qualitative aspects about the states of the CTMC can be expressed. In state 1 the atomic propositions *off* indicates the state of the system where it is turned off. Similarly state 2, 3, 4 and 5 indicate the *sleep*, *idle*, *receive* and *transmit* states of the system. In states 4 and 5 the system is either transmitting or receiving in which case it is considered to be busy indicated by the atomic proposition *busy.*

Chapter 3

Markov Reward Model (MRM)

This chapter presents the definition of Markov Reward Model *and a logic for the specification of properties over such MRM. The definition of MRM is presented in section* 3.1*. Section* 3.2 *describes paths in an MRM. In section* 3.3*, the probability measure defined over paths in MRM is described. Section* 3.4 *presents state occupation probabilities. In Section* 3.5*, the definition of performability and its interpretation over MRMs is presented. A logic for the specification of properties over such MRM is presented in section* 3.6*. Characterization of steady-state and of transient measures is described in section* 3.7 *and in section* 3.8 *respectively.*

3.1 Markov Reward Model

A Markov Reward Model is formally defined as:

Definition 3.1 (Markov Reward Model (MRM)) *A Markov Reward Model (MRM) $\mathcal{M}$ is a three-tuple $((S, \boldsymbol{R}, Label), \rho, \iota)$ where $(S, \boldsymbol{R}, Label)$ is the underlying labeled CTMC $\mathcal{C}$, $\rho : S \longrightarrow \mathbb{R}_{\geq 0}$ is the state reward structure, and $\iota : S \times S \longrightarrow \mathbb{R}_{\geq 0}$ is the impulse reward structure such that if $\boldsymbol{R}_{s,s} > 0$ then $\iota(s, s) = 0$.*

An MRM is a labeled CTMC augmented with state reward and impulse reward structures. The state reward structure is a function ρ that assigns to each state $s \in S$ a reward $\rho(s)$ such that if t time-units are spent in state s, a reward of $\rho(s) \cdot t$ is acquired. The rewards that are defined in the state reward structure can be interpreted in various ways. They can be regarded as the gain or benefit acquired by staying in some state and they can also be regarded as the cost spent by staying in some state.

The impulse reward structure, on the other hand, is a function ι that assigns to each transition from s to s', where $s, s' \in S$ and $\boldsymbol{R}_{s,s'} > 0$, a reward $\iota(s, s')$ such that if the transition from s to s' occurs, a reward of $\iota(s, s')$ is acquired. Similar to the state reward structure, the impulse reward structure can be interpreted in various ways. An impulse reward can be considered as the cost of taking a transition or the gain that is acquired by taking the transition.

Example 3.1 *Back to the WaveLAN modem example in the previous chapter, the given labeled CTMC can be extended to an MRM by augmenting it by state reward and impulse reward structures.*

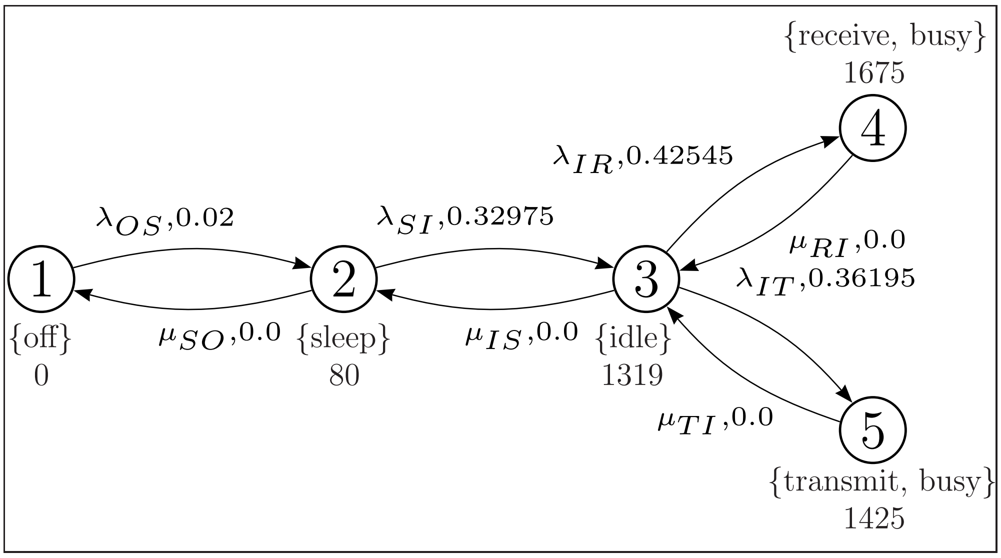


Figure 3.1: WaveLAN Modem Modeled as an MRM

The state reward and impulse reward structures in this example are interpreted as cost in terms of energy consumption. A WaveLAN modem, as described in [Pau01], typically consumes 1675 *mW while transmitting,* 1425 *mW while receiving,* 1319 *mW while being idle and* 80 *mW while in sleep mode. This information is used to develop the state reward structure as follows:*

$$\begin{aligned} &\rho(1) = 0\ mW, && \rho(2) = 80\ mW, \\ &\rho(3) = 1319\ mW, && \rho(4) = 1675\ mW, \\ &\rho(5) = 1425\ mW. \end{aligned}$$

It is also described that transitions from sleep to idle take 250 *μs with the power consumption of the idle state. It is also noted that* 254 *μs is required before the payload is transmitted. It is further assumed that the same duration of time is required before a payload can be received. Similarly the transition from off to sleep is assumed to take* 250 *μs with the power consumption of the sleep state. These observations are interpreted as impulse rewards, thus for defining the impulse rewards the transitions are assumed to be instantaneous. Hence the impulse reward structure is as follows:*

$$\begin{aligned} &\iota(1,2) = 80 \cdot 250 \cdot 10^{-6} = 0.02\ mJ, && \iota(3,4) = 1675 \cdot 254 \cdot 10^{-6} = 0.425\,45\ mJ, \\ &\iota(2,1) = 0\ mJ, && \iota(4,3) = 0\ mJ, \\ &\iota(2,3) = 1319 \cdot 250 \cdot 10^{-6} = 0.32975\ mJ, && \iota(3,5) = 1425 \cdot 254 \cdot 10^{-6} = 0.361\,95\ mJ, \\ &\iota(3,2) = 0\ mJ, && \iota(5,3) = 0\ mJ. \end{aligned}$$

Figure 3.1 presents the labeled directed graph of the MRM described in the above example. In addition to the states and transitions that are represented by vertices and edges respectively, the labeled directed graph of an MRM also contains representations of label sets, state rewards and impulse rewards. The label set and state reward of a state are placed, in that order, near the vertex representing the state. The impulse reward of a transition is placed near the edge representing the transition after the rate of the transition.

Definition 3.2 (Absorbing State in an MRM) *A state $s \in S$ in an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ is absorbing if and only if $\boldsymbol{R}_{s,s'} = 0$ for all $s' \in S$.*

3.2 Paths in MRM

During its running time, an MRM makes transitions from states to states. Informally, the sequence of the states that the MRM resides is called a *path*. In this section the concept of a path in MRMs is formalized. This concept provides a tool to compute several measures in an MRM.

Definition 3.3 (Path in MRMs) *An* infinite *path σ in MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots$, with $s_i \in S$ and $t_i \in \mathbb{R}_{>0}$ is the amount of time spent in state s_i where $i \in \mathbb{N}$ and $\boldsymbol{R}_{s_i,s_{i+1}} > 0$ for all i. A* finite *path σ in $\mathcal{M}$ is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots \xrightarrow{t_{n-1}} s_n$, such that s_n is an absorbing state and $\boldsymbol{R}_{s_i,s_{i+1}} > 0$ for all $i < n$. For finite paths that end in s_n, $t_n = \infty$. For infinite path σ, let $\sigma[i] = s_i, i \in \mathbb{N}$. For finite path σ that ends in s_n, $\sigma[i] = s_i$ is only defined for $i \leq n$. The last state in the finite path σ (the (n+1)-st state) is referred to as $last(\sigma)$. Two additional functions are defined over paths in MRMs:*

The state being occupied at time t on path σ is defined as:

$$\sigma@t = \sigma[i] \Leftrightarrow \sum_{j=0}^{i-1} t_j < t \ \wedge \ \sum_{j=0}^{i} t_j \geq t.$$

Rewards accumulated at time t in a path σ is a function $y_\sigma : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$ such that:

$$y_\sigma(t) = \rho(\sigma[i]) \cdot \left(t - \sum_{j=0}^{i-1} t_j \right) + \sum_{j=0}^{i-1} \rho(\sigma[j]) \cdot t_j + \sum_{j=0}^{i-1} \iota(\sigma[j], \sigma[j+1]) \text{ where } \sigma@t = \sigma[i].$$

Thus, there are two types of paths: infinite and finite paths. An infinite path contains infinitely many transitions. In finite paths, there are a finite number of transitions, but the last state in the path is absorbing and an infinite amount of time is spent in this state. $Paths^{\mathcal{M}}$ is the set of finite and infinite paths in the MRM, $Paths^{\mathcal{M}}(s)$ is the set of finite and infinite paths in the MRM that start in state s, thus $\forall \sigma \in Paths^{\mathcal{M}}(s).\sigma[0] = s$.

For instance, path:

$$\sigma = s_1 \xrightarrow{0.5} s_3 \xrightarrow{100} s_2 \xrightarrow{2} s_1 \xrightarrow{17} s_4,$$

is a finite path. Hence, the last state in the path, s_4, is absorbing and an infinite amount of time is spent in this state. In this path $\sigma[2]$ is the (2+1)-st state in the path, namely s_2. Similarly $\sigma[0] = s_1$, $\sigma[1] = s_3$, $\sigma[3] = s_1$ and $\sigma[4] = s_4$. $\sigma[i]$ is only defined for $i \leq 4$. $last(\sigma) = s_4$, since s_4 is the last state in the path.

A certain state can be identified to be occupied at a given time t on path σ assuming that time started from the initial state on the path. Then the i-th state on path σ is occupied at time t if the sum of the residence time of all states before the i-th state is less than t and the sum of the residence time of all states including the i-th state is greater than t.

In a similar fashion, rewards accumulated at time t in a path σ, $y_\sigma(t)$, can be obtained from the definition of path in MRMs, given that the state being occupied at time t is $\sigma@t = \sigma[i]$. The first addendum indicates the amount of reward accumulated due to residence in the i-th state until time t. The second addendum indicates the sum of rate rewards accumulated due to residence in states prior to the visit to the i-th state. The third addendum indicates the impulse rewards accumulated on the path σ.

Example 3.2 *Consider an infinite path in the WaveLAN modem as an MRM example as follows:*

$$\sigma = 1 \xrightarrow{10} 2 \xrightarrow{4} 3 \xrightarrow{2} 4 \xrightarrow{3.75} 3 \xrightarrow{1} 5 \xrightarrow{2.5} 3 \xrightarrow{5} \cdots .$$

The state that is occupied at time 21.75 *in path* σ *is* $\sigma@21.75 = \sigma[5] = 5$ *since:*

$$\sum_{j=0}^{4} t_j = 20.75 < 21.75 \ \wedge \ \sum_{j=0}^{5} t_j = 23.25 \geq 21.75.$$

Further, the rewards accumulated at that time is $y_\sigma(21.75)$ *where:*

$$\begin{aligned} y_\sigma(21.75) &= \rho(\sigma[5]) \cdot \left(21.75 - \sum_{j=0}^{4} t_j\right) + \sum_{j=0}^{4} \rho(\sigma[j]) \cdot t_j + \sum_{j=0}^{4} \iota(\sigma[j], \sigma[j+1]) \\ &= \rho(5) \cdot (21.75 - 20.75) + \rho(1) \cdot 10 + \rho(2) \cdot 4 + \rho(3) \cdot 2 + \rho(4) \cdot 3.75 \\ &\quad + \rho(3) \cdot 1 + \iota(1,2) + \iota(2,3) + \iota(3,4) + \iota(4,3) + \iota(3,5) \\ &= 11983.25\ mW \cdot s\ + 1.13715\ mJ = 11984.38715\ mJ. \end{aligned}$$

Thus, in this path, after 21.75 *seconds,* 11984.38715 mJ *of energy is consumed. The path* σ *is an element of* $Paths^{\mathcal{M}}(1)$.

3.3 Probability of Paths

To be able to define and evaluate measures that are defined over MRMs it is necessary to obtain a probability measure over paths in the MRM. Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$, every state $s \in S$ and $s = s_0$ gives a probability measure over all paths that originate in state s_0 by Borel Space construction [Bai00]. Let $s_0, \ldots, s_k \in S$ with $\boldsymbol{R}_{s_i, s_{i+1}} > 0$ for $0 \leq i < k$ and $I_0, \ldots, I_{k-1}$ be non-empty intervals in $\mathbb{R}_{\geq 0}$. Then, $C(s_0, I_0, \ldots, I_{k-1}, s_k)$ denotes the cylinder set consisting of all paths $\sigma \in Paths^{\mathcal{M}}(s)$ such that $\sigma[i] = s_i$ and $t_i \in I_i$ for $0 \leq i < k$. Let $\mathcal{F}(Paths^{\mathcal{M}}(s))$ be the smallest σ-algebra defined over $Paths^{\mathcal{M}}(s)$ which contains all the sets $C(s, I_0, \ldots, I_{k-1}, s_k)$ with $s_0, \ldots, s_k$ ranging over all state sequences and $s = s_0$ where $\boldsymbol{R}_{s_i, s_{i+1}} > 0$ for $0 \leq i < k$ and $I_0, \ldots, I_{k-1}$ ranges over all non-empty intervals in $\mathbb{R}_{\geq 0}$. Then the probability measure over $\mathcal{F}(Paths^{\mathcal{M}}(s))$ is defined by induction over k as follows:

$$\begin{aligned}
\Pr\{C(s_0)\} &= 1 \text{ for } k = 0, \\
\Pr\{C(s, I_0, \ldots, I_{k-1}, s_k, I', s')\} &= \Pr\{C(s, I_0, \ldots, I_{k-1}, s_k)\} \cdot P(s_k, s') \cdot (e^{-E(s_k) \cdot a} - e^{-E(s_k) \cdot b}),
\end{aligned}$$

where $a = \inf(I')$ and $b = \sup(I')$ and the probability of taking the transition from state s_k to state s' within the time-interval I' is:

$$\int_{I'} P(s_k, s') \cdot E(s_k) \cdot e^{-E(s_k) \cdot t} dt = P(s_k, s') \cdot (e^{-E(s_k) \cdot a} - e^{-E(s_k) \cdot b}),$$

where $E(s_k) \cdot e^{-E(s_k) \cdot t}$ is the probability density function of the state residence time in state s_k at time t.

3.4 Transient and Steady-State Probability

Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ with the underlying CTMC $\mathcal{C}$. In chapter 2 for such CTMC $\mathcal{C}$ two kinds of state probabilities have been defined viz. the steady-state probability and the transient probability. The transient probability is the state occupation probability after a bounded time-interval t has elapsed. Let $\pi(s, s', t)$ represent the probability of starting in state $s \in S$ and reaching state $s' \in S$ within t time-units. By the definition of probability of paths this probability is formally defined as:

$$\pi^{\mathcal{M}}(s, s', t) = \Pr\{\sigma \in Paths^{\mathcal{M}}(s) | \sigma@t = s'\}.$$

The steady-state analysis aims at determining the state occupation probabilities after an unbounded observation interval has elapsed. Let $\pi^{\mathcal{M}}(s, s')$ represent the steady-state probability of starting in state $s \in S$ and reaching state $s' \in S$. This measure is given by:

$$\pi(s, s') = \lim_{t \to \infty} \pi^{\mathcal{M}}(s, s', t).$$

When the underlying CTMC is strongly-connected then this probability does not depend on the initial-probability distribution and this probability is referred to as $\pi(s')$. This measure can also be defined for a set of states $S' \subseteq S$:

$$\pi(s, S') = \sum\nolimits_{s' \in S'} \pi(s, s').$$

3.5 MRM and Performability

The definition and evaluation based on MRMs was initiated due to the need for simultaneous analysis of *performance* and *dependability* of computer systems, also referred to as *performability* [Mey95]. Whilst performance in computer systems is the efficacy by which the system delivers services under the assumption that the services delivered conform to the specification, dependability is the ability of the system to deliver services that conform to the specification. Although these two issues may be analyzed independently, simultaneous analysis becomes imperative when the performance of the system *degrades* under the presence of behavior that does not conform to the specification.

The Performability Measure Y is a random variable; its specification includes a *utilization period* T, which is an interval of the time base and an *accomplishment set* A, in which Y takes its values. The performability of the system $Perf(B)$ relative to a specified Y where $B \subseteq A$, is defined to be $Perf(B) = \Pr\{Y \in B\}$. An example of such an accomplishment set is the set of all non-negative real numbers $\mathbb{R}_{\geq 0}$. The performability measure for a subset of the accomplishment set is then the probability that the performability variable interpreted over the stochastic process takes values in the subset. For instance, a subset of the accomplishment set $\mathbb{R}_{\geq 0}$ is $[0, y]$, $y \in \mathbb{R}_{\geq 0}$.

Definition 3.4 (The Performability Measure $Y(I)$) *The performability of a system, modeled by an MRM in the utilization interval I in the time base such that the accumulated reward (accomplishment) is in J, is $Perf(J) = \Pr\{Y(I) \in J\}$.*

The accumulated reward over an MRM in a finite interval can be considered to be a performability variable. Then performability over an MRM for a finite interval $[0, t]$ and for a set $[0, y]$, $y \in \mathbb{R}_{\geq 0}$ is defined as the probability that the accumulated reward is in $[0, y]$ over a utilization interval $[0, t]$ and the performability measure is then defined to be $Perf(\leq r) = \Pr\{Y(t) \leq r\}$.

3.6 A Logic for MRMs (CSRL)

Once the model of the system as an MRM is available some means for the expression of requirements over the model is necessary. In many practical situations one is also concerned with the ability to impose requirements over the temporal ordering of certain events. Such requirements or properties to be interpreted over an MRM are specified in the *Continuous Stochastic Reward Logic (CSRL)* [Bai00, Hav02].

The definition of such a logic provides the means for expressing requirements of the system being modeled as an MRM. The definition encapsulates two subcomponents viz. the syntax and the semantics. The syntax of the logic defines precisely what constitutes a CSRL formula. It provides the means to specify properties of the system modeled as an MRM. However, the syntax alone does not explain the interpretation of these formulas over the system. This interpretation is explained by the semantics. The syntax and semantics of CSRL over MRMs with both state and impulse reward structures are as follows:

3.6.1 Syntax of CSRL

The definition of the syntax of CSRL is concerned with the development of means by which properties of systems being modeled as MRMs can be expressed. In CSRL two kinds of formulas are distinguished viz. *state formulas* and *path formulas*. Formulas whose validity is investigated given a state are referred to as state formulas while formulas whose validity is investigated given a path are said to be path formulas. The first step is the definition of a set of atomic propositions, AP. The definition of such a set AP determines the set of qualitative properties of the system that can be expressed. Given a set AP the following definition presents the formulas that can be expressed in CSRL:

Definition 3.5 (Syntax of Continuous Stochastic Reward Logic (CSRL)) *Let $p \in [0,1]$ be a real number, $\trianglelefteq \in \{<, \leq, \geq, >\}$ a comparison operator and I and J intervals of non-negative real numbers. The syntax of CSRL formulas over the set of atomic propositions AP is defined as follows:*

`tt` *is a state formula,*
Each $a \in AP$ is a state formula,
If Φ and Ψ are state formulas then $\neg\Phi$ and $\Phi \vee \Psi$ are state formulas,
If Φ is a state formula then $\mathcal{S}_{\trianglelefteq p}(\Phi)$ is a state formula,
If φ is a path formula then is $\mathcal{P}_{\trianglelefteq p}(\varphi)$ a state formula,
If Φ and Ψ are state formulas then $\mathcal{X}^I_J\Phi$ and $\Phi\mathcal{U}^I_J\Psi$ are path formulas.

Interval I is a timing constraint, while interval J is a bound for the accumulated reward. $\mathcal{X}$ is called ne($\mathcal{X}$)t operator, while $\mathcal{U}$ is called ($\mathcal{U}$)ntil operator. These formulas can be distinguished as follows:

State Formulas

1. The boolean operators are as in propositional logic. Other boolean operators $\wedge$ and $\Rightarrow$ can be derived as follows: $\Phi \wedge \Psi = \neg(\neg\Phi \vee \neg\Psi)$ and $\Phi \Rightarrow \Psi = \neg\Phi \vee \Psi$.

2. The measure $\mathcal{S}_{\unlhd p}(\Phi)$ is referred to as the *steady-state measure*. The state formula $\mathcal{S}_{\unlhd p}(\Phi)$ asserts that the steady-state probability for the set of Φ-states meets the bound $\unrhd p$.

3. The measure $\mathcal{P}_{\unlhd p}(\varphi)$ is referred to as the *transient probability measure*. The state formula $\mathcal{P}_{\unlhd p}(\varphi)$ asserts that the probability measure of paths satisfying φ meets the bound $\unrhd p$.

Path Formulas

1. The measure $\mathcal{X}_J^I\Phi$ asserts that a transition is made to a Φ-state at time $t \in I$ such that the accumulated reward until time t, $r \in J$.

2. The measure $\Phi\mathcal{U}_J^I\Psi$ asserts that Ψ-formula is satisfied at some future time $t \in I$ such that the accumulated reward until time t, $r \in J$ and the Φ-formula is satisfied at all instants before t.

3. Additional formulae are defined by direct consequence of the syntax of CSRL: $\Diamond_J^I\Phi = \texttt{tt}\mathcal{U}_J^I\Phi$ and $\mathcal{P}_{\unlhd p}(\Box_J^I\Phi) = \neg\mathcal{P}_{\unlhd p}(\Diamond_J^I\neg\Phi)$.

Example 3.3 *Some interesting measures of the WaveLAN modem example that can be expressed in CSRL are as follows:*

- *The probability is more than* 0.5 *that the system is either transmitting or receiving after a certain duration of time has elapsed. It is assumed that the system has only* 50 *J of energy. Then after an observation interval of* 10 *minutes is allowed to elapse this property is:* $\mathcal{P}_{>0.5}(\texttt{tt}\mathcal{U}_{[0,50]}^{[0,600]} busy)$.

- *Energy consumption in WaveLAN can be significantly reduced if the interface spends most of the time in sleep mode. One way to improve this is by imposing requirements on the ability of the system to reach the sleep state from busy or idle state within a certain time-duration. This property can be specified as: the probability is more than* 0.8 *that the system reaches the sleep state from busy or idle state before a certain duration of time has elapsed. If it is assumed that the system has only* 50 *J of energy and the duration allowed is* 10 *seconds this property in CSRL is:* $\mathcal{P}_{>0.8}((busy \vee idle)\mathcal{U}_{[0,50]}^{[0,10]} sleep)$.

- *Nested measures for instance the CSRL formula,* $\mathcal{P}_{>0.8}(\mathcal{X}(\mathcal{P}_{>0.5}\mathcal{X}_{[0,50]}^{[0,10]} sleep)))$ *specifies that a state is reached after one transition from the starting state with a probability of at least* 0.8 *from which it is possible to reach the sleep state with a probability of more than* 0.5 *in one transition within* 10 *seconds and expending less than* 50 *J of energy.*

3.6.2 Semantics of CSRL

The syntax of CSRL provides the rules to construct valid CSRL formulas or to check whether a given formula is a valid CSRL formula. The interpretation of all operators in CSRL is given by its semantics. The semantics of formulas in CSRL is defined by means of a satisfaction relation $\models$ between a state s and a state formula Φ, and between a path σ and a path formula φ. A satisfaction relation is called valid iff a state formula is valid in a state or a path formula is valid for a path. The semantics of CSRL is defined as follows:

Definition 3.6 (Semantics of Continuous Stochastic Reward Logic (CSRL)) *CSRL formulas are interpreted over an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ by a satisfaction relation $\models$, which is defined for state formulas and path formulas as follows:*

$s \models \mathtt{tt}\ \forall s \in S$,
$s \models a \Leftrightarrow a \in Label(s)$,
$s \models \neg\Phi \Leftrightarrow\ \neg(s \models \Phi)$,
$s \models \Phi \vee \Psi \Leftrightarrow s \models \Phi \vee s \models \Psi$,
$s \models \mathcal{S}_{\trianglelefteq p}(\Phi) \Leftrightarrow \pi(s, Sat(\Phi)) \trianglelefteq p$,
$s \models \mathcal{P}_{\trianglelefteq p}(\varphi) \Leftrightarrow \Pr\{\sigma \in Paths(s) | \sigma \models \varphi\} \trianglelefteq p$,
$\sigma \models \mathcal{X}^I_J \Phi \Leftrightarrow \sigma[1]$ *is defined* $\wedge\ \sigma[1] \models \Phi \wedge t_0 \in I \wedge y_\sigma(t_0) \in J$,
$\sigma \models \Phi \mathcal{U}^I_J \Psi \Leftrightarrow \exists t \in I.(\sigma@t \models \Psi \wedge (\forall t' \in [0, t).\sigma@t' \models \Phi) \wedge y_\sigma(t) \in J)$.

Recall that t_0 is the residence time in the initial state in the given path σ. The interpretation of these formulas is as follows:

State Formulas

If state $s \models \Phi$ then it is said that state s satisfies the state formula Φ. The interpretation of state formulas is as follows:

1. The interpretation of the boolean operators is as in propositional logic.
2. $s \models \mathcal{S}_{\trianglelefteq p}(\Phi)$ iff the steady-state probability $\pi(s, Sat(\Phi))$ for the set of Φ-states starting from the initial state s, meets the bound $\trianglerighteq p$.
3. $s \models \mathcal{P}_{\trianglelefteq p}(\varphi)$ iff the probability measure of paths satisfying φ meets the bound $\trianglerighteq p$. The definition of the probability of such paths is derived from the Borel Space construction [Bai03] and is measurable.

Path Formulas

1. $\sigma \models \mathcal{X}^I_J \Phi$ iff a transition is made to a Φ-state at time $t_0 \in I$ such that the accumulated reward until time t_0, $y_\sigma(t_0) \in J$.

2. $\sigma \vDash \Phi \mathcal{U}_J^I \Psi$ iff the Ψ-formula is satisfied at some future time $t \in I$, $\sigma @ t \vDash \Psi$, such that the accumulated reward until time t, $y_\sigma(t) \in J$ and the Φ-formula is satisfied at all instants before t. Note that once the Ψ-formula is satisfied on path σ the future behavior of the path is irrelevant as far as the validity of the path formula is concerned.

Let $P^{\mathcal{M}}(s, \varphi)$ denote the probability with which the given path formula φ is satisfied starting from state $s \in S$:

$$P^{\mathcal{M}}(s, \varphi) = \Pr\{\sigma \in Paths(s) | \sigma \vDash \varphi\}.$$

Example 3.4 *Consider the formula specified in the previous example for the WaveLAN modem to verify that the probability is more than* 0.5 *that the system is either transmitting or receiving after a certain duration of time has elapsed:* $\mathcal{P}_{>0.5}(\mathtt{tt}\mathcal{U}_{[0,50]}^{[0,600]} busy)$. *Further consider the following path:*

$$\sigma = 1 \xrightarrow{100} 2 \xrightarrow{40} 3 \xrightarrow{20} 4 \xrightarrow{37.5} 3 \xrightarrow{10} 5 \xrightarrow{25} 3 \xrightarrow{50} \cdots .$$

From figure 3.1, $\sigma[3] \vDash busy$ *and all previous states in path* σ *satisfy* $\mathtt{tt}$ *and* $\exists t = 160 \in [0, 600]$ *such that:*

$$\sigma \vDash \Phi \mathcal{U}_J^I \Psi \Leftrightarrow (\sigma @ 160 \vDash busy \wedge (\forall t' \in [0, 160). \sigma @ t' \vDash \mathtt{tt}) \wedge y_\sigma(160) = 29.581 \in [0, 50]).$$

Consequently, it can be concluded that $\sigma \vDash \mathtt{tt}\mathcal{U}_{[0,600]}^{[0,50]} busy$.

3.7 Steady-State Measures

Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$, a starting state $s_0 \in S$ and a formula of the form $\mathcal{S}_{\trianglelefteq p}(\Phi)$, the question whether s_0 satisfies the given formula, can be answered by using analysis in [Bai03]. In this procedure first the set of states $Sat(\Phi) = \{s \in S | s \vDash \Phi\}$ is found. Subsequently, the set of states satisfying $\mathcal{S}_{\trianglelefteq p}(\Phi)$ namely $Sat(\mathcal{S}_{\trianglelefteq p}(\Phi))$ is determined, by distinguishing the following two cases:

1. **A strongly connected CTMC**: When the underlying CTMC of $\mathcal{M}$ is strongly connected then a standard CTMC steady-state analysis by the solution of a linear system of equations suffices:

$$s_0 \in Sat(\mathcal{S}_{\trianglelefteq p}(\Phi)) \text{ iff } \sum_{s' \in Sat(\Phi)} \pi(s_0, s') \trianglelefteq p. \qquad (3.1)$$

2. **CTMC is not strongly connected**: When the underlying CTMC of $\mathcal{M}$ is not strongly connected then a graph analysis of the CTMC is performed to obtain the bottom-strongly connected components (BSCCs). Now each

BSCC is a strongly connected CTMC. Consequently standard CTMC steady-state analysis can be used for each BSCC. The question whether s_0 satisfies the given formula can be answered by the following analysis:

$$s_0 \in Sat(\mathcal{S}_{\unlhd p}(\Phi)) \text{ iff } \sum_{B} \left(P(s_0, \Diamond B) \cdot \sum_{s' \in B \cap Sat(\Phi)} \pi^B(s') \right) \unlhd p. \tag{3.2}$$

$P(s_0, \Diamond B)$ is the probability of satisfying the formula $(\Diamond B)$ or $(tt\mathcal{U}B)$ starting from state s_0. Characterization of this measure is given by equation (3.8). Note that equation (3.2) reduces to equation (3.1) when the CTMC is a BSCC itself.

Example 3.5 *Consider the example CTMC in figure 3.2 and the computation involved to check $\mathcal{S}_{\geq 0.3}(b)$ for state s_1. A graph analysis reveals that there are two BSCCs in the example viz. $B1 = \{s_3, s_4\}$ and $B2 = \{s_5\}$.*

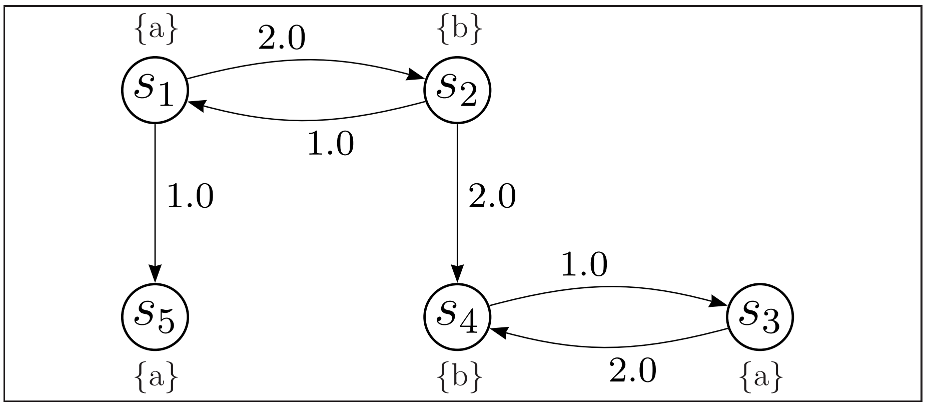


Figure 3.2: BSCCs in Steady-State Analysis

The first step is to determine the set of states that satisfy b-formula. As s_4 is the only b-state belonging to one of the BSCCs hence by the application of equation (3.2), the steady-state probability $\pi(s_1, Sat(b))$ is:

$$\pi(s_1, Sat(b)) = \Big(P(s_1, \Diamond B1) \cdot \pi^{B1}(s_4) \Big),$$

where $P(s_1, \Diamond B1)$ can be determined by a solution for:

$$P(s_1, \Diamond B1) = \frac{2}{3} \cdot P(s_2, \Diamond B1) \text{ and } P(s_2, \Diamond B1) = \frac{2}{3} + \frac{1}{3} \cdot P(s_1, \Diamond B1).$$

Hence $P(s_1, \Diamond B1) = \frac{4}{7}$. For $\pi^{B1}(s_4)$ the following equations have to be solved:

$$2 \cdot \pi^{B1}(s_3) - \pi^{B1}(s_4) = 0 \text{ and } \pi^{B1}(s_3) + \pi^{B1}(s_4) = 1,$$

yielding $\pi^{B1}(s_4) = \frac{2}{3}$. Hence, $\pi(s_1, Sat(b)) = \frac{4}{7} \cdot \frac{2}{3} = \frac{8}{21}$ and $s_1 \models \mathcal{S}_{\geq 0.3}(b)$.

3.8 Transient Measures

Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, \mathit{Label}), \rho, \iota)$, to resolve whether the transient probability measure of the form $s \vDash \mathcal{P}_{\trianglelefteq p}(\varphi)$ is satisfied by the given model and state s, it is first necessary to determine the probability with which the given path formula φ is satisfied. Define:

$$\begin{aligned} K(s) &= \{x \in I | \rho(s) \cdot x \in J\}, \text{ and} \\ K(s, s') &= \{x \in I | (\rho(s) \cdot x + \iota(s, s')) \in J\}, \end{aligned}$$

for closed intervals I and J. $K(s)$ is an interval of time in I such that the state-rewards accumulated by being resident in state s for any length of time in $K(s)$ is in J. $K(s, s')$ is an interval of time in I such that the sum of the state-rewards accumulated by being resident in state s for any length of time in $K(s, s')$ and the impulse reward acquired by transition from state s to s' is in J. The probability density of moving from state s to s' within x time-units is $P(s, s', x) = P(s, s') \cdot E(s) \cdot e^{-E(s) \cdot x}$. Hence the probability of leaving state s within the interval I such that both the time and reward bounds are met after the outgoing transition has taken place is:

$$P_J^I(s) = \sum_{s' \in S} \int_{K(s,s')} P(s, s') \cdot E(s) \cdot e^{-E(s) \cdot x} dx, \tag{3.3}$$

where $s, s' \in S$. For instance consider the case where unbounded reward is allowed to be accumulated:

$$\begin{aligned} P_{[0,\infty)}^{[0,t]}(s) &= \sum_{s' \in S} \int_0^t P(s, s') \cdot E(s) \cdot e^{-E(s) \cdot x} dx = \sum_{s' \in S} P(s, s') \cdot (-e^{-E(s) \cdot x}) \Big|_0^t \\ &= \sum_{s' \in S} P(s, s') \cdot (1 - e^{-E(s) \cdot t}) = (1 - e^{-E(s) \cdot t}) \cdot \sum_{s' \in S} P(s, s') = (1 - e^{-E(s) \cdot t}). \end{aligned}$$

3.8.1 Ne$\mathcal{X}$t Formula

For formula $s \vDash \mathcal{P}_{\trianglelefteq p}(\mathcal{X}_J^I \Phi)$, let $P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi)$ be the probability of satisfying the formula $(\mathcal{X}_J^I \Phi)$ starting from state s. From equation (3.3), $P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi)$ is:

$$P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi) = \sum_{s' \vDash \Phi} P(s, s') \cdot (e^{-E(s) \cdot \inf(K(s,s'))} - e^{-E(s) \cdot \sup(K(s,s'))}). \tag{3.4}$$

This characterization suggests that the algorithm to compute $s \vDash \mathcal{P}_{\trianglelefteq p}(\mathcal{X}_J^I \Phi)$ proceeds by first obtaining all states that satisfy Φ. Subsequently, $P^{\mathcal{M}}(s, \mathcal{X}_J^I \Phi)$ is determined by the use of equation (3.4) and this probability is compared with the specified probability p. Special cases exist for the next formulas when $J = [0, \infty)$ and $I = [0, \infty)$:

$$P^{\mathcal{M}}(s, \mathcal{X}_{[0,\infty)}^{[0,\infty)} \Phi) = P^{\mathcal{M}}(s, \mathcal{X} \Phi) = \sum_{s' \vDash \Phi} P(s, s'). \tag{3.5}$$

3.8.2 $\mathcal{U}$ntil Formula

For formula $s \vDash \mathcal{P}_{\trianglelefteq p}(\Phi \mathcal{U}_J^I \Psi)$, let $P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi)$ be the probability of satisfying the formula $(\Phi \mathcal{U}_J^I \Psi)$ starting from state s. For time-reward bounded until formula $\mathcal{P}_{\trianglelefteq p}(\Phi \mathcal{U}_J^I \Psi)$, $P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi)$ is characterized by a fixed-point equation. First let $L \ominus y = \{l - y | l \in L \wedge l \geq y\}$, then $P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi)$ is the least solution of the following set of equations:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}_J^I \Psi) = \begin{cases} 1, \text{ if } s \vDash \neg\Phi \wedge \Psi \text{ and } \inf(I) = 0 \text{ and } \inf(J) = 0, \\ \sum_{s' \in S} \int_0^{\sup(K(s,s'))} P(s, s', x) \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U}^{I \ominus x}_{J \ominus (\rho(s) \cdot x + \iota(s,s'))} \Psi) dx, \\ \hfill \text{if } s \vDash \Phi \wedge \neg\Psi, \\ e^{-E(s) \cdot \inf(K(s))} \\ + \sum_{s' \in S} \int_0^{\inf(K(s,s'))} P(s, s', x) \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U}^{I \ominus x}_{J \ominus (\rho(s) \cdot x + \iota(s,s'))} \Psi) dx, \\ \hfill \text{if } s \vDash \Phi \wedge \Psi, \\ 0, \text{ otherwise.} \end{cases} \tag{3.6}$$

The justification of this characterization is as follows:

1. $s \vDash \Psi$ and $\inf(I) = 0$ and $\inf(J) = 0$: Since s satisfies Ψ and $\inf(I) = 0$ and $\inf(J) = 0$ then all paths starting from state s satisfy the formula and consequently the probability is 1.

2. $s \vDash \Phi \wedge \neg\Psi$: If s satisfies $(\Phi \wedge \neg\Psi)$ then the probability of reaching a Ψ-state from state s within the interval I and by accumulating reward $r \in J$ is the probability of reaching a direct successor s' within x time-units, ($x \leq \sup(I)$ and $(\rho(s) \cdot x + \iota(s, s')) \leq \sup(J)$, that is $x \leq \sup(K(s, s'))$) multiplied with the probability of reaching a Ψ-state from state s' within the interval $I \ominus x$ and by accumulating reward $(r - (\rho(s) \cdot x + \iota(s, s')))$.

3. $s \vDash \Phi \wedge \Psi$: If s satisfies $(\Phi \wedge \Psi)$ then the path formula is satisfied if the state s is not left for $\inf(K(s))$ time-units. Alternatively, state s should be left before $\inf(K(s, s'))$ time-units have elapsed in which case the probability is defined as in case 2. Note that by definition $\inf(K(s, s')) \leq \inf(K(s))$.

Whilst the system of equations described above completely characterizes the until formula some trivial cases still exist. If $J = [0, \infty)$ and $I = [0, t]$ for $t \in \mathbb{R}_{\geq 0}$ then the characterization of the path formula is given by the least solution of the following set of equations:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}^{[0,t]}_{[0,\infty)} \Psi) = \begin{cases} 1, \text{ if } s \vDash \Psi, \\ \int_0^t \sum_{s' \in S} P(s, s', x) \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U}^{[0, t \ominus x]} \Psi) dx, \\ \hfill \text{if } s \vDash \Phi \wedge \neg\Psi, \\ 0, \text{ otherwise.} \end{cases} \tag{3.7}$$

which corresponds to the characterization in [Bai03].

If $J = [0, \infty)$ and $I = [0, \infty)$ then:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U}^{[0,\infty)}_{[0,\infty)} \Psi) = P^{\mathcal{M}}(s, \Phi \mathcal{U} \Psi). \tag{3.8}$$

The solution to the RHS of equation (3.8) is the least solution of the following set of linear equations:

$$P^{\mathcal{M}}(s, \Phi \mathcal{U} \Psi) = \begin{cases} 1, \text{ if } s \models \Psi, \\ \sum_{s' \in S} P(s, s') \cdot P^{\mathcal{M}}(s', \Phi \mathcal{U} \Psi), \\ \qquad\qquad\qquad \text{if } s \models \Phi \wedge \neg \Psi, \\ 0, \text{ otherwise.} \end{cases}$$

The solution to equation (3.8) can be computed by the solution of linear equations by standard means such as Gaussian elimination or iterative strategies such as the Gauss-Seidel method.

Example 3.6 *Consider the WaveLAN modem model of example 3.1. In this example it is demonstrated as to how the value of the formula $P^{\mathcal{M}}(3, idle \mathcal{U}^{[0,2]}_{[0,2000]} busy)$ can be computed using the characterization for until formula as presented in equation (3.6).*

By condition (2) in equation (3.6):

$$\begin{aligned} P^{\mathcal{M}}(3, idle \mathcal{U}^{[0,2]}_{[0,2000]} busy) &= \int_0^a \lambda_{IR} \cdot e^{-E(3) \cdot x} \cdot P^{\mathcal{M}}(4, idle \mathcal{U}^{[0,2-x]}_{[0,2000-1319 \cdot x - 0.42545]} busy) dx \\ &+ \int_0^b \lambda_{IT} \cdot e^{-E(3) \cdot x} \cdot P^{\mathcal{M}}(5, idle \mathcal{U}^{[0,2-x]}_{[0,2000-1319 \cdot x - 0.36195]} busy) dx, \end{aligned}$$

where $E(3) = (\lambda_{IR} + \lambda_{IT} + \mu_{IS})$, $a = \frac{2000-0.42545}{1319}$, $b = \frac{2000-0.36195}{1319}$. By condition (1) in equation (3.6):

$$P^{\mathcal{M}}(4, idle \mathcal{U}^{[0,2-x]}_{[0,2000-1319 \cdot x - 0.42545]} busy) = P^{\mathcal{M}}(5, idle \mathcal{U}^{[0,2-x]}_{[0,2000-1319 \cdot x - 0.36195]} busy) = 1,$$

hence:

$$P^{\mathcal{M}}(3, idle \mathcal{U}^{[0,2]}_{[0,2000]} busy) = \int_0^a \lambda_{IR} \cdot e^{-E(3) \cdot x} dx + \int_0^b \lambda_{IT} \cdot e^{-E(3) \cdot x} dx$$

Assume rates $\lambda_{IR} = 1.5\ hr.^{-1}, \lambda_{IT} = 0.75\ hr.^{-1}, \mu_{IS} = 12\ hr.^{-1}$, then:

$$\begin{aligned} P^{\mathcal{M}}(3, idle \mathcal{U}^{[0,2]}_{[0,2000]} busy) &= \int_0^a 1.5 \cdot e^{-14.25 \cdot x} dx + \int_0^b 0.75 \cdot e^{-14.25 \cdot x} dx \\ &= 0.10526 + 5.2632 \cdot 10^{-2} = 0.15789. \end{aligned}$$

Although full characterization of until formulas is presented in equation (3.6), the solution of this equation by standard means for the solution of integral functions is a difficult task and susceptible to numerical instability. In chapter 4 methods for the computation of these measures without the need for evaluating the integral are presented.

Chapter 4

Model Checking MRMs

This chapter presents algorithms for model checking Markov Reward Models. An overview of the model checking procedure is provided in section 4.1. Section 4.2 describes algorithms for model checking steady-state formulas. In section 4.3, algorithms for model checking transient probability measures are described. Section 4.4 presents numerical methods for model checking transient probability measures. Applications of the numerical methods for model checking transient probability measures are presented in section 4.5 and in section 4.6.

4.1 The Model Checking Procedure

For model checking MRMs, given a system modeled as an MRM and a formula expressed in CSRL, it has to be ascertained whether the formula is valid for the MRM. Initially the formula is parsed and its sub-formulas are determined. Then a post-order recursive traversal of the parse-tree is carried out to determine the values of the sub-formulas. The value of a formula is the set of states that satisfy the formula. Such a set of states that satisfy a formula Φ is referred to as $Sat(\Phi)$. An algorithm to obtain $Sat(\Phi)$ is algorithm 4.1.

Algorithm 4.1 SatisfyStateFormula

```
SatisfyStateFormula(StateFormula $\Phi$): SetOfStates
  if $\Phi = \texttt{tt}$ then return $S$
  if $\Phi \in AP$ then return $\{s|\Phi \in Label(s)\}$
  if $\Phi = \neg\Phi_1$ then return $S-$SatisfyStateFormula($\Phi_1$)
  if $\Phi = \Phi_1 \vee \Phi_2$ then return SatisfyStateFormula($\Phi_1$) $\cup$ SatisfyStateFormula($\Phi_2$)
  if $\Phi = \mathcal{S}_{\trianglelefteq p}(\Phi_1)$ then return SatisfySteady($p, \trianglelefteq$,SatisfyStateFormula($\Phi_1$))
  if $\Phi = \mathcal{P}_{\trianglelefteq p}(\mathcal{X}_J^I \Phi_1)$ then return SatisfyNext($p, \trianglelefteq$,SatisfyStateFormula($\Phi_1$)$, I, J$)
  if $\Phi = \mathcal{P}_{\trianglelefteq p}(\Phi_1 \mathcal{U}_J^I \Phi_2)$ then
    return SatisfyUntil($p, \trianglelefteq$,SatisfyStateFormula($\Phi_1$),SatisfyStateFormula($\Phi_2$)$, I, J$)
end SatisfyStateFormula
```

Algorithms to determine SatisfySteady, SatisfyNext and SatisfyUntil are presented in sections 4.2, 4.3.1 and 4.3.2 respectively.

4.2 Model Checking Steady-State Operator

The state formula $\mathcal{S}_{\unlhd p}(\Phi)$, given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ and a starting state $s \in S$, asserts that the steady-state probability for the set of Φ-states meets the bound $\unlhd p$. As has been illustrated in chapter 3 when the underlying CTMC of $\mathcal{M}$ is strongly connected then a standard CTMC steady-state analysis by the solution of a linear system of equations (3.1) suffices.

When the underlying CTMC of $\mathcal{M}$ is not strongly connected then first a graph analysis is performed to determine all the bottom-strongly connected components (BSCCs). Every BSCC is a strongly connected component and standard CTMC analysis to determine the steady-state probability for each BSCC can be carried out. For finding maximal SCCs (MSCC) standard algorithms are available and can be readily modified to obtain BSCCs. To find BSCCs the Tarjan's algorithm for obtaining MSCCs as presented in [Nuu93] is considered. Subsequently, every one of the MSCCs has to be further analyzed to determine if it is a BSCC. For this purpose it has to ascertained that all the successors of states belonging to a MSCC belong to the MSCC too.

Consider algorithm 4.2. The Tarjan's algorithm has two procedures viz. procedure visit and procedure getBSCC. Procedure getBSCC applies procedure visit to all the states that have not been visited. Procedure visit initially considers the state being visited as the *root* of a component. Subsequently all the successor states of the state being visited are considered. If one of the successor states is not visited yet then it is visited too. Once all the successor states have been visited the root of the state being visited is the minimum of its initial root and the roots of its successor states which are not already in a component. Note that minimum here refers to the dfs (depth-first search) order in which the states are visited. If at this point the root of the state being visited is still the original dfs order of the state then a MSCC is said to have been detected and the states in the MSCC are on the top of the stack.

With reference to the detection of the BSCC two cases arise during the evolution of the algorithm. When a certain state is being visited either it can transit to a state which has not been visited or to a state which has already been visited. If it can transit to a state that has already been visited either this new state is in the present stack or is part of a detected component and not in the stack. Consequently if a transition to a state already detected to be in a component is possible then the present component being detected can never be a BSCC. If a new state is visited then it would either be a part of the present component being detected or it would be a new component. If the new state is found to be in a new component then a transition to another component from the present state is possible. Consequently the component being visited can never be a BSCC.

To detect BSCCs, the procedure visit has been augmented with a boolean vector *reachSCC* which records whether it is possible to reach a strongly connected component for every state. If a MSCC can be reached from a state say s then the MSCC in which s belongs could never be a BSCC.

The computational cost for Tarjan's algorithm is $\mathcal{O}(M+N)$ time where M is the number of non-zero elements in the rate matrix and N is the number of states. The time-complexity of the modified Tarjan's algorithm for detecting BSCC is $\mathcal{O}(M+N)$ too.

Algorithm 4.2 Bottom-Strongly Connected Component

```
getBSCC(StateSpace $S$, RateMatrix $\boldsymbol{R}$): ListOfBSCC
  initialize $stack$, bool [$|S|$] $reachSCC$, integer[$|S|$] $root$
  initialize bool[$|S|$] $inComponent$, ListOfBSCC $bscclist$, integer $dfsorder = 0$
  for each $s \in S$ such that $root[s] = 0$
    visit($s$)  /* root[s]=0 $\Rightarrow$ $s$ is not visited */
  end for
  return bsccList
end getBSCC

visit(State $s$): void
  increment $dfsorder$
  initialize integer $remorder = dfsorder$
  initialize BSCC $newbscc$, bool $reachSCC = false$
  $root[s] = dfsorder$, $inComponent[s] = false$, $visited[s] = true$
  push($s$, $stack$)
  for each $s' \in S$ such that $\boldsymbol{R}_{s,s'} > 0$
    if $root[s'] = 0$ then visit($s'$)
    if $\neg inComponent[s']$ then $root[s] = \min(root[s], root[s'])$
      else $reachSCC[s] = true$
  end for
  if $root[s] = remorder$ then
    do
      $s' =$ pop($stack$)
      $reachSCC = reachSCC \vee reachSCC[s']$
      $newbscc$.add($s'$)
      $inComponent[s'] = true$
    while $s \neq s'$
    if $\neg reachSCC$
      then $bscclist$.add($newbscc$)
  end if
end visit
```

Once the BSCC are determined, a steady-state analysis by standard means by the solution of the system of linear equations by the Gauss-Seidel method is used for each BSCC. Subsequently, the probability of reaching each BSCC is determined by the evaluation of equation (3.8). The final result is obtained by the use of equation (3.2) for obtaining $Sat(\mathcal{S}_{\trianglelefteq p}(\Phi))$. An algorithm to determine $Sat(\mathcal{S}_{\trianglelefteq p}(\Phi))$ is algorithm 4.3. Note that it is not necessary to distinguish the two cases explicitly.

Algorithm 4.3 SatisfySteady

```
SatisfySteady(ProbabilityBound $p$, ComparisonOperator $\trianglelefteq$, SetOfStates $Sat_\Phi$): SetOfStates
  initialize SetOfStates $SatSteady = \emptyset$
  initialize ListOfBSCC $bsccList$ = getBSCC($S, \boldsymbol{R}$)
  if $|bsccList| = 1$ and $bsccList(1) = S$ then
    if $\sum_{s' \in Sat_\Phi} \pi(s') \trianglelefteq p$ then $SatSteady = S$
  else
    for each $s \in S$
      if $\sum_{B \in bsccList} \left( P(s, \Diamond B) \cdot \sum_{s' \in B \cap Sat_\Phi} \pi^B(s') \right) \trianglelefteq p$
        then $SatSteady = SatSteady \cup \{s\}$
    end for
  end if
  return $SatSteady$
end SatisfySteady
```

Note that $P(s, \Diamond B)$ is determined by associating an extra atomic proposition atB to all the states in BSCC B. Subsequently a system of linear equations (3.8) has to be solved to obtain $P(s, \Diamond atB)$. By doing this, $P(s, \Diamond B)\ \forall s \in S$ is obtained. S refers to the state space and $\boldsymbol{R}$ is the rate matrix of the MRM.

In conclusion the computation involved for the evaluation of the steady-state measure requires first to determine the BSCCs which requires $\mathcal{O}(M + N)$ time where M is the number of non-zero elements in the rate matrix and N is the size of state space. Subsequently $|B|$ number of equations need to be solved for one BSCC, considering all the BSCC the total number of equations in the worst case is N. Then N number of equations need to be solved to determine steady-state probability of all BSCCs. The solution of linear equations requires $\mathcal{O}(N^3)$ time if direct-methods such as Gauss Elimination are used. If the Gauss-Seidel method is used for solving linear equations then the time complexity depends on the convergence of the Gauss-Seidel method.

4.3 Model Checking Transient Measures

4.3.1 Next Formulas

The characterization given in equation (3.4) can be used to develop an algorithm to model check formulas that contain the next operator. Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$, a starting state $s_0 \in S$ and a formula of the form $\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^I_J \Phi)$, the question whether s_0 satisfies the given formula, can be answered by evaluating $P^{\mathcal{M}}(s_0, \mathcal{X}^I_J \Phi)$ which is the probability of satisfying the formula $(\mathcal{X}^I_J \Phi)$ starting from state s_0.

The algorithm for checking next formula in this case will proceed by computing the probability of satisfying next formula for each state s. This consists of determining $K(s, s')$ for each state s' which satisfy Φ-formula. Recall that $K(s, s')$ is an interval of time in I such that the sum of the state-rewards accumulated by being resident in state s for any length of time in $K(s, s')$ and the impulse reward acquired by transition from state s to s' is in J.

For each s' the probability of making a transition from state s to s' within $K(s, s')$ time-units is computed. The sum of all such transition probabilities for all s' is then compared to the probability bound. Those states that satisfy the probability bound are the subset of state space that satisfy the transient probability measure with next formula. Algorithm 4.4 finds the subset of state space that satisfies the transient probability measure with next operator.

Algorithm 4.4 SatisfyNext

```
SatisfyNext(ProbabilityBound $p$, ComparisonOperator $\trianglelefteq$, SetOfStates $Sat_\Phi$, TimeInterval $I$, RewardInterval $J$): SetOfStates
  initialize SetOfStates $SatNext = \emptyset$
  for each $s \in S$
    $Prob = 0$
    for each $s' \in Sat_\Phi$
      compute $K(s, s')$
      $Prob = Prob + P(s, s') \cdot (e^{-E(s) \cdot \inf(K(s,s'))} - e^{-E(s) \cdot \sup(K(s,s'))})$
    end for
    if $Prob \trianglelefteq p$ then $SatNext = SatNext \cup \{s\}$
  end for
  return $SatNext$
end SatisfyNext
```

For transient probability measure with next formula involving the special cases when $I = [0, \infty)$ and $J = [0, \infty)$, equation (3.5) can be utilized to compute the probability of satisfying next formula for a particular initial state.

4.3.2 Until Formulas

The characterization given in equation (3.6) can be used to develop an algorithm to model check formulas that contain the until operator. However, the evaluation of the integral is susceptible to numerical instability. In this section methods to transform the MRM are presented such that standard algorithms for the analysis of MRMs can be used. For model checking until formulas firstly as in [Hav02] the following types of properties are distinguished:

P0: $\mathcal{P}_{\leq p}(\Phi \mathcal{U} \Psi)$: This formula asserts that the probability of reaching a Ψ-state by residing in only Φ-states is at most the probability specified. Values for this property are obtained by a solution of linear system of equations (3.8).

P1: $\mathcal{P}_{\leq p}(\Phi \mathcal{U}^{[0,t]} \Psi)$: This formula asserts that the probability of reaching a Ψ-state within t time-units and by residing in only Φ-states is less or equal to the probability specified. This measure is characterized by equation (3.7). Efficient methods to compute the value of this property are presented in [Bai03].

P2: $\mathcal{P}_{\leq p}(\Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi)$: This formula asserts that the probability of reaching a Ψ-state within t time-units and by accumulating rewards at most the reward bound r and by residing in only Φ-states is less or equal to the probability specified. To resolve the values of properties of this type when only state reward rates are present numerical solutions are presented in [Hav02]. When the underlying model has impulse rewards too this value is characterized by equation (3.6).

Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$, a starting state $s_0 \in S$ and a formula of the form $\mathcal{P}_{\trianglelefteq p}(\Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi)$, the question whether s_0 satisfies the given formula can be answered by first evaluating $P^{\mathcal{M}}(s_0, \Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi)$ which is the probability of satisfying the formula $(\Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi)$ starting from state s_0.

The following theorems provide a method to transform the MRM so that standard algorithms for the evaluation of MRMs can be used to compute $P^{\mathcal{M}}(s_0, \Phi \mathcal{U}_{[0,r]}^{[0,t]} \Psi)$. This transformation involves making certain states in the model absorbing and assigning zero impulse rewards to all transitions originating from such states, and assigning zero state reward rate to the state. Formally the procedure to make this transformation is captured in definition 4.1:

Definition 4.1 *For MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$ and CSRL state formula Φ, let $\mathcal{M}[\Phi] = ((S, \boldsymbol{R}', Label), \rho', \iota')$ be an MRM such that:*

$$\begin{aligned} \boldsymbol{R}'_{s,s'} &= \begin{cases} \boldsymbol{R}_{s,s'} \text{ if } s \not\models \Phi \\ 0 \text{ otherwise} \end{cases}, \\ \rho'(s) &= \begin{cases} \rho(s) \text{ if } s \not\models \Phi \\ 0 \text{ otherwise} \end{cases}, \\ \iota'(s,s') &= \begin{cases} \iota(s,s') \text{ if } s \not\models \Phi \\ 0 \text{ otherwise} \end{cases}, \end{aligned}$$

for all $s, s' \in S$.

$\mathcal{M}[\Phi]$ is thus obtained from $\mathcal{M}$ by making all Φ-states absorbing and equipping these states with zero rewards. Note that applying definition 4.1 for Φ-states followed by applying it for Ψ-states is the same as applying it for the union of Φ-states and Ψ-states, thus $\mathcal{M}[\Phi][\Psi] = \mathcal{M}[\Phi \vee \Psi]$. In the following sections the term *absorbing*-states is used to refer to states which are transformed as illustrated in definition 4.1

Definition 4.1 allows us to develop an algorithm to make states which satisfy certain formula absorbing. In the subsequent sections MakeAbsorbing(MRM, SetOfStates) refers to such an algorithm. The following example shows the result obtained after making certain states absorbing in the WaveLAN modem example.

Example 4.1 *Let $\mathcal{M}$ be the MRM given in the example 3.1 presented in chapter 3 and let $\Phi =$ busy. Then $\mathcal{M}$[busy] is an MRM obtained from $\mathcal{M}$ by making all* busy*-states absorbing. The MRM $\mathcal{M}$[busy] is given in figure 4.1.*

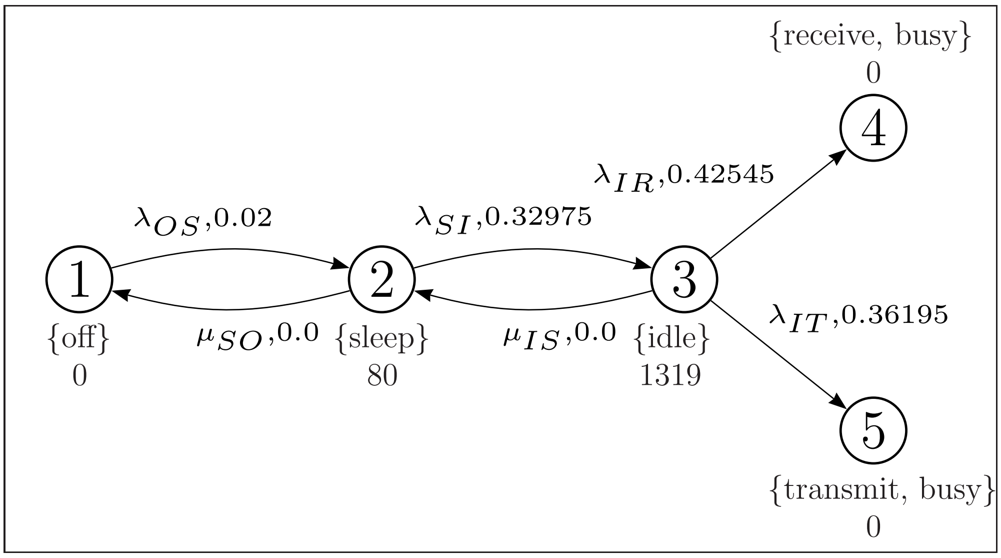


Figure 4.1: WaveLAN Modem Model where *busy*-states are made absorbing

Having formalized the notion of making states absorbing in MRM, the following theorem maintains that the probability of satisfying the formula $(\Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ starting from state s is equal to the probability of satisfying the formula $(\mathtt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ starting from state s once all $(\neg\Phi \vee \Psi)$-states are made absorbing:

Theorem 4.1 *Given an MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$, then for $s \in S$:*

$$P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{J}\Psi) = P^{\mathcal{M}[\neg\Phi\vee\Psi]}(s, \mathtt{tt}\mathcal{U}^{[t,t]}_{J}\Psi).$$

Proof *From the semantics of until formula:*

$$\begin{aligned} & P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{J}\Psi) \\ = \ & \Pr\{\sigma \in Paths^{\mathcal{M}}(s) | \exists x \in [0,t].(\sigma@x \vDash \Psi \wedge (\forall y \in [0,x).\sigma@y \vDash \Phi) \wedge y_\sigma(x) \in J)\}. \end{aligned}$$

In $\mathcal{M}[\Psi]$ all the Ψ-states are made absorbing, state reward assigned to these states is 0. Hence once a Ψ-state is reached in $\mathcal{M}[\Psi]$ it cannot be left and no reward is earned any further. Accordingly:

$$P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{J}\Psi) = \Pr\{\sigma \in Paths^{\mathcal{M}[\Psi]}(s) | (\sigma@t \vDash \Psi \wedge (\forall y \in [0,t).\sigma@y \vDash \Phi \vee \Psi) \wedge y_\sigma(t) \in J)\}.$$

$\mathcal{M}[\Psi][\neg\Phi \wedge \neg\Psi]$ is obtained from $\mathcal{M}[\Psi]$ in which all the $(\neg\Phi \wedge \neg\Psi)$-states are made absorbing, state reward assigned to these states is 0. Hence once a $(\neg\Phi \vee \Psi)$-state is reached it cannot be left and no reward is earned any further. Accordingly:

$$P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{J}\Psi) = \Pr\{\sigma \in Paths^{\mathcal{M}[\neg\Phi\vee\Psi]}(s) | \sigma@t \vDash \Psi \wedge y_\sigma(t) \in J\}.$$

From the semantics of until formula:

$$P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{J}\Psi) = P^{\mathcal{M}[\neg\Phi\vee\Psi]}(s, \mathtt{tt}\mathcal{U}^{[t,t]}_{J}\Psi).$$

□

Hence satisfying $(\Phi\mathcal{U}_J\Psi)$ within time $[0,t]$ can be checked by investigating whether formula $(\Diamond_J\Psi)$ is satisfied at exactly time t if all $(\neg\Phi\vee\Psi)$-states in the MRM are made absorbing.

A similar procedure can be employed to compute the probability of satisfying the formula $(\Phi\mathcal{U}_J^{[t,t]}\Psi)$ starting from state s, namely the probability of satisfying the formula at exactly time t, while accumulating J rewards starting from state s, as given by the following theorem:

Theorem 4.2 *Given an MRM $\mathcal{M}=((S,\boldsymbol{R},Label),\rho,\iota)$, a CSRL formula $\mathcal{P}_{\trianglelefteq p}(\Phi\mathcal{U}_J^{[t,t]}\Psi)$ and $\Psi\Rightarrow\Phi$, then for $s\in S$:*

$$P^{\mathcal{M}}(s,\Phi\mathcal{U}_J^{[t,t]}\Psi)=P^{\mathcal{M}[\neg\Phi\wedge\neg\Psi]}(s,\mathtt{tt}\mathcal{U}_J^{[t,t]}\Psi).$$

Proof *From the semantics of until formula:*

$$\begin{aligned}&P^{\mathcal{M}}(s,\Phi\mathcal{U}_J^{[t,t]}\Psi)\\ =\;&\Pr\{\sigma\in Paths^{\mathcal{M}}(s)\,|\,(\sigma@t\models\Psi\wedge(\forall y\in[0,t).\sigma@y\models\Phi)\wedge y_\sigma(t)\in J)\}.\end{aligned}$$

$\mathcal{M}[\neg\Phi\wedge\neg\Psi]$ is obtained from $\mathcal{M}$ in which all the $(\neg\Phi\wedge\neg\Psi)$-states are made absorbing and state reward assigned to these states is 0. Consequently, once a $(\neg\Phi\wedge\neg\Psi)$-state is reached, it cannot be left.

However Ψ-states are not made absorbing hence they could possibly be left and another Ψ-state could be reached at time t. Since $\Psi\Rightarrow\Phi$ hence the condition $\sigma@y\models\Phi$ is no longer necessary:

$$P^{\mathcal{M}}(s,\Phi\mathcal{U}_J^{[t,t]}\Psi)=\Pr\{\sigma\in Paths^{\mathcal{M}[\neg\Phi\wedge\neg\Psi]}(s)\,|\,(\sigma@t\models\Psi\wedge y_\sigma(t)\in J)\}.$$

From the semantics of until formula:

$$P^{\mathcal{M}}(s,\Phi\mathcal{U}_J^{[t,t]}\Psi)=P^{\mathcal{M}[\neg\Phi\wedge\neg\Psi]}(s,\mathtt{tt}\mathcal{U}_J^{[t,t]}\Psi).$$

$\square$

Hence satisfying $(\Phi\mathcal{U}_J\Psi)$ at time t can be checked by investigating whether formula $(\Diamond_J\Psi)$ is satisfied at time t if all $(\neg\Phi\wedge\neg\Psi)$-states in the model are made absorbing, provided that $\Psi\Rightarrow\Phi$.

The following theorem maintains that the probability measure $P^{\mathcal{M}}(s,\mathtt{tt}\mathcal{U}_{[0,r]}^{[t,t]}\Psi)$ is equal to the joint probability of residing in Ψ-states at time t <u>and</u> accumulating rewards less or equal to r at time t.

Theorem 4.3 *Given an MRM $\mathcal{M}=((S,\boldsymbol{R},Label),\rho,\iota)$, then for $s\in S$:*

$P^{\mathcal{M}}(s,\mathtt{tt}\mathcal{U}_{[0,r]}^{[t,t]}\Psi)=\Pr\{Y(t)\leq r,X(t)\models\Psi\}$ *interpreted over $\mathcal{M}$ and* $\underline{p}_s(0)=1$.

Proof *From the semantics of until formula:*

$$P^{\mathcal{M}}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi) = \Pr\{\sigma \in Paths^{\mathcal{M}}(s) | \sigma@t \vDash \Psi \wedge y_\sigma(t) \in [0,r]\}.$$

Let process $\{X(t), Y(t)\}$ be a two-dimensional stochastic process representing the state $X(t)$ being occupied at time t and the reward accumulated Y(t) at time t. Then the probability that a Ψ-state is occupied at time t and the reward accumulated at time t is in $[0,r]$ in the stochastic process is: $\Pr\{Y(t) \le r, X(t) \vDash \Psi\}$.

Since the starting state is s:

$$P^{\mathcal{M}}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi) = \Pr\{Y(t) \le r, X(t) \vDash \Psi\} \textit{ interpreted over } \mathcal{M} \textit{ and } \underline{p}_s(0) = 1.$$

□

The theorems presented above allow us to develop an algorithm to find the subset of state space that satisfies transient probability measure with until operator restricted in time-interval $[0,t]$ and reward-interval $[0,r]$, as follows:

Algorithm 4.5 SatisfyUntil

```
SatisfyUntil(ProbabilityBound $p$, ComparisonOperator $\trianglelefteq$, SetOfStates $Sat_\Phi$, SetOfStates $Sat_\Psi$, TimeUpperBound $t$, RewardUpperBound $r$): SetOfStates
  initialize SetOfStates $SatUntil = \emptyset$
  initialize MRM $\mathcal{M}' =$ MakeAbsorbing($\mathcal{M}, Sat^c_\Phi \cup Sat_\Psi$)
  for each $s \in S$
    if $P^{\mathcal{M}'}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi) \trianglelefteq p$ then $SatUntil = SatUntil \cup \{s\}$
  end for
  return $SatUntil$
end SatisfyUntil
```

Note that $P^{\mathcal{M}'}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi) = \Pr\{Y(t) \le r, X(t) \in Sat_\Psi\}$ interpreted over $\mathcal{M}'$. The algorithm proceeds by first making all $(\neg\Phi \vee \Psi)$-states absorbing. For each state in the state space the probability measure $P^{\mathcal{M}}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ is computed and the probability is compared with the probability bound. Those states that satisfy the probability bound are the subset of state space that satisfy the transient probability measure with until formula. Two numerical methods to compute $P^{\mathcal{M}}(s, \texttt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ are provided in the following section.

4.4 Numerical Methods

4.4.1 Discretization

This method is based on the discretization algorithm in [Tij02] for transient performability measures. It discretizes both the time interval and the accumulated reward as multiples of the same step size d where d is chosen such that the probability of more than one transition in the MRM in an interval of length d is negligible. Using discretization method, the probability of accumulating less or equal to r reward in an MRM is given by:

$$\Pr\{Y(t) \leq r\} = \sum_{s\in S} \sum_{k=1}^{k=R} F^T(s,k) \cdot d,$$

where $R = \frac{r}{d}$, and $T = \frac{t}{d}$.

$F^T(s,k)$ is the probability of being in state s, at discretized time T and with accumulated discretized reward k. The initial condition $F^1(s,k)$ is given by:

$$F^1(s,k) = \begin{cases} 1/d & \text{if } (s,k) = (s_0, \rho(s_0)), \\ 0 & \text{otherwise.} \end{cases}$$

For subsequent intervals the following recursive scheme is used:

$$F^{j+1}(s,k) = F^j(s, k-\rho(s)) \cdot (1 - E(s)\cdot d) + \sum_{s'\in S} F^j(s', k-\rho(s') - \frac{\iota(s',s)}{d}) \cdot \boldsymbol{R}_{s',s} \cdot d.$$

For the MRM to be in state s at the $(j+1)$-st time-instant either the MRM was in state s in the j-th time-instant and remained there for d time-units without traversing a self-loop (the first summand) or it was in state s' and has moved to state s in that period (the second sum). Given that the accumulated reward at the $(j+1)$-st time-instant is k the accumulated reward in the j-th time-instant is approximated by $(k - \rho(s))$ in the first summand and it is $(k - \rho(s') - \frac{\iota(s',s)}{d})$ in the second summand.

Note that the discretization method requires non-negative state reward rates. If the method is implemented by using matrices to store F^j and F^{j+1}, then it is also necessary to have integer state reward rates. Rational state reward rates can be scaled to integer value. If the state reward rates are scaled then the reward bound in the formula should be appropriately scaled too.

The discretization method can be used to compute $P^{\mathcal{M}}(s, \mathtt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$, which is explained in section 4.5. In section 4.5 the complexity of the method is described.

4.4.2 Uniformization for MRMs

Uniformization is one of the most widely applied strategies to evaluate transient measures that are defined over an MRM [Sil94]. This section presents methods to evaluate transient measures defined over an MRM by using uniformization. A uniformized MRM can be obtained by uniformizing (section 2.4.1) the underlying CTMC of the MRM. The following is the definition of the uniformized MRM:

Definition 4.2 (Uniformized MRM) *Let $\mathcal{M}$ be an MRM defined as $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$. A uniformized MRM $\mathcal{M}^u = (S, \boldsymbol{P}, \Lambda, Label, \rho, \iota)$ is obtained by uniformizing the underlying CTMC of the MRM and Λ is the rate of the Poisson process $\{N_t : t \geq 0\}$ associated with $\mathcal{M}^u$.*

Example 4.2 *Consider example 3.1 in chapter 3. Assume that the rates* $\lambda_{OS} = 0.1$*,* $\lambda_{SI} = 5$*,* $\lambda_{IR} = 1.5$*,* $\lambda_{IT} = 0.75$*,* $\mu_{SO} = 0.05$*,* $\mu_{IS} = 12$*,* $\mu_{RI} = 10$*,* $\mu_{TI} = 15$ hr.$^{-1}$*. The rate matrix is as follows:*

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0.1 & 0 & 0 & 0 \\ 0.05 & 0 & 5 & 0 & 0 \\ 0 & 12 & 0 & 1.5 & 0.75 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 \end{bmatrix}.$$

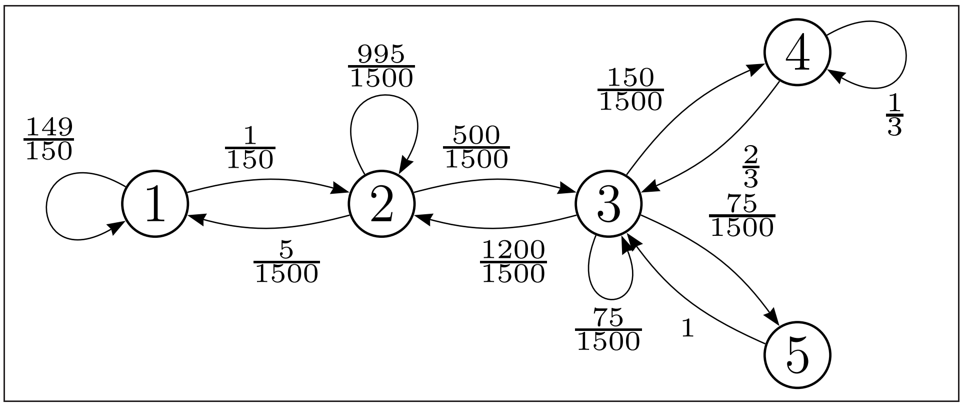


Figure 4.2: WaveLAN Modem Model after Uniformization

Hence the total rate of taking an outgoing transition from each state ($E(s)$) *is* $E(1) = 0.1$*,* $E(2) = 5.05$*,* $E(3) = 14.25$*,* $E(4) = 10$ *and* $E(5) = 15$*. Applying uniformization to this model with Poisson process' rate* $\Lambda = \max_i(E(s)) = 15$ *the* 1*-step probabilities matrix of the uniformized process is given by:*

$$\boldsymbol{P} = \boldsymbol{I} + \frac{\boldsymbol{R} - Diag(E)}{\Lambda} = \begin{bmatrix} \frac{149}{150} & \frac{1}{150} & 0 & 0 & 0 \\ \frac{5}{1500} & \frac{995}{1500} & \frac{500}{1500} & 0 & 0 \\ 0 & \frac{1200}{1500} & \frac{75}{1500} & \frac{150}{1500} & \frac{75}{1500} \\ 0 & 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Figure 4.2 presents the state-transition diagram of the uniformized model. The labeling, state and impulse reward information is ignored in the figure.

Now, it is possible to evaluate transient measures that are defined over MRM $\mathcal{M}$ by using the uniformized MRM $\mathcal{M}^u$. Consider the evolution of the MRM approximated by the uniformized MRM. The MRM resides in a state for a certain duration of time given by the transition times in the Poisson process that is associated with the uniformized MRM. After such a transition the uniformized MRM transits to the present state with a certain probability if more residence in the state is necessary to approximate the behavior of the original MRM. Alternatively it transits to another state with certain probability and so on.

The amount of reward accumulated in the MRM depends on the nature of such evolution trajectories (i.e. rewards of the states involved). Hence the computation of the distribution of accumulated rewards requires a mechanism to characterize such trajectories on the basis of rewards. Consider a trajectory of evolution of the MRM. The trajectory consists of a sequence of states in the uniformized MRM. Every residence corresponds to the time for a transition in the Poisson process. Such a trajectory of length n could be characterized with a vector of $n+1$ numbers corresponding to the reward assigned to states in the trajectory.

Alternatively, the path may contain many states which have identical rewards assigned to them consequently it suffices to characterize a path with a vector of length equal to the number of distinct rewards assigned to states in the MRM. Every position in the vector now characterizes the number of residences in states with a certain reward. Using such a characterization of paths on the basis of the rewards assigned to states in individual trajectories the PDF of the performability measure $Y(t)$, the reward accumulated within $[0,t]$ over $\mathcal{M}$, was derived by de Souza e Silva & Gail in [Sil94] and can be expressed as follows:

$$\Pr\{Y(t) \leq r\} = \sum_{n=0}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} \cdot \sum_{\forall \underline{k}} \Pr\{\underline{k}|n\} \cdot \Pr\{Y(t) \leq r|n, \underline{k}\} \tag{4.1}$$

$\underline{k} = \langle k_1, k_2, k_3, \cdots, k_{K+1}\rangle$ is a vector where $K+1$ is the number of distinct state rewards $r_1 > r_2 > r_3 > \cdots > r_{K+1} \geq 0$ in $\mathcal{M}$ and k_i is the number of entrances to some state with state reward r_i such that $\sum_{i=1}^{K+1} k_i = n+1$.

If impulse rewards are present in the underlying MRM then every trajectory is further characterized by an additional vector. The length of such a vector is equal to the number of distinct impulse rewards that are assigned to transitions. Every position in this new vector now characterizes the number of times that transitions with a certain impulse reward occur in a trajectory. By characterizing every trajectory by a combination of such two vectors to incorporate impulse rewards, Qureshi & Sanders [Qur94] generalized equation (4.1) to obtain:

$$\Pr\{Y(t) \leq r\} = \sum_{n=0}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} \cdot \sum_{\forall \underline{k}} \sum_{\forall \underline{j}} \Pr\{\underline{k}, \underline{j}|n\} \cdot \Pr\{Y(t) \leq r|n, \underline{k}, \underline{j}\} \tag{4.2}$$

$\underline{j} = \langle j_1, j_2, j_3, \cdots, j_J\rangle$ is a vector where J is the number of distinct impulse rewards $i_1 > i_2 > i_3 > \cdots > i_J \geq 0$ in $\mathcal{M}$ and j_i is the number of occurrences of transitions with impulse reward i_i such that $\sum_{i=1}^{J} j_i = n$.

Depth Truncation

The exact computation of equation (4.2) requires the computation of an infinite sum. Typically, this computation has to be truncated at some depth N. This is referred to as *depth truncation*. If depth truncation is employed then equation (4.2) can be written as:

$$\Pr\{Y(t) \leq r\} \approx \sum_{n=0}^{N} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} \cdot \sum_{\forall \underline{k}} \sum_{\forall \underline{j}} \Pr\{\underline{k}, \underline{j}|n\} \cdot \Pr\{Y(t) \leq r|n, \underline{k}, \underline{j}\} \tag{4.3}$$

Note that the only difference with equation (4.2) is that the upper bound of the summation is now fixed.

Path Truncation

Truncation of the computation of equation (4.2) can alternatively be achieved by conditioning on paths on the basis of the probability of occurrence of the path.

Definition 4.3 (Path in Uniformized MRM) *A path in a uniformized MRM $\mathcal{M}^u = (S, \boldsymbol{P}, \Lambda, Label, \rho, \iota)$ is a sequence of states $s_0 \longrightarrow s_1 \longrightarrow \cdots$, with $s_i \in S$ and $\boldsymbol{P}_{s_i,s_{i+1}} > 0\ \forall i \geq 0$. $Paths^{\mathcal{M}^u}$ is the set of all paths in a uniformized MRM; $Paths^{\mathcal{M}^u}(s)$ is the set of paths in the uniformized MRM where for all $\sigma \in Paths^{\mathcal{M}^u}(s), \sigma[0] = s$.*

A finite path σ in $\mathcal{M}^u$ is a prefix of a path in $\mathcal{M}^u$ of length $n \geq 0$. A finite path is of the form $s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n$ is called a Finite Path in Uniformized MRM. *$FPaths^{\mathcal{M}^u}$ be the set of finite paths is $\mathcal{M}^u$. $FPaths^{\mathcal{M}^u}(s)$ be the set of finite paths is $\mathcal{M}^u$ where for all $\sigma \in FPaths^{\mathcal{M}^u}(s), \sigma[0] = s$. Let $last(\sigma)$ be the final state of a finite path σ, i.e. $last(\sigma) = s_n$.*

Definition 4.4 (Probability of a Finite Path) *The probability of a finite path $\sigma = s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n$ in a uniformized MRM $\mathcal{M}^u = (S, \boldsymbol{P}, \Lambda, Label, \rho, \iota)$ given the initial probability distribution $\underline{p}(0)$ is:*

$$P(\sigma) = \underline{p}_{s_0}(0) \cdot \boldsymbol{P}_{s_0,s_1} \cdot \boldsymbol{P}_{s_1,s_2} \cdot \boldsymbol{P}_{s_2,s_3} \cdot \cdots \cdot \boldsymbol{P}_{s_{n-1},s_n}.$$

Definition 4.5 (Probability of a Finite Path at time t) *The probability of being in the final state at time t in a finite path $\sigma = s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n$ in a uniformized MRM $\mathcal{M}^u = (S, \boldsymbol{P}, \Lambda, Label, \rho, \iota)$ is:*

$$P(\sigma, t) = \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} \cdot P(\sigma) \text{ where } n = |\sigma|.$$

Definition 4.6 (Path Truncation) *The truncation of paths can be performed in such a way that the paths with probability less than a certain truncation probability w are ignored in the computation. This truncation is called* path truncation. *The set of truncated paths given w is:*

$$TP(\mathcal{M}, w, s, t) = \{\sigma \in FPaths^{\mathcal{M}^u}(s) | 0 < w \leq P(\sigma, t) \leq 1\}.$$

If path truncation is employed then the PDF of the performability measure $Y(t)$ over $\mathcal{M}$, derived by Qureshi & Sanders [Qur96] is:

$$\Pr\{Y(t) \leq r\} \approx \sum_{\sigma \in TP(\mathcal{M},w,s,t)} P(\sigma, t) \cdot \Pr\{Y(t) \leq r | \sigma\}. \tag{4.4}$$

Note that several paths may be represented by the same value of $(n, \underline{k}, \underline{j})$. For instance if several states have the same state reward and transitions have identical impulse rewards. This implies that once the value of the conditional probability expressed above is computed it can be stored to avoid recomputation. Alternatively the computation of the conditional probabilities can be initiated after the probability of all relevant paths is known.

4.5 Discretization Applied to Until Formulas

From theorems 4.1 and 4.3, the probability measure $P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ is given by:

$$P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi) = \Pr\{Y(t) \leq r, X(t) \models \Psi\} \text{ over } \mathcal{M}[\neg\Phi \vee \Psi] \text{ and } \underline{p}_s(0) = 1.$$

Based on these theorems, the discretization method can be used to compute $P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$, by first making all $(\neg\Phi \vee \Psi)$-states in the model absorbing and then restricting the first sum to be performed only on the states that satisfy Ψ, namely:

$$\Pr\{Y(t) \leq r, X(t) \models \Psi\} = \sum_{s' \models \Psi} \sum_{k=1}^{k=R} F^T(s', k) \cdot d,$$

where $R = \frac{r}{d}, T = \frac{t}{d}$, and $F^T(s, k)$ is computed as had been demonstrated in section 4.4.1.

Algorithm 4.6 is an algorithm to compute the probability measure $P^{\mathcal{M}}(s, \Phi\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ using discretization method. It is assumed that prior to the invocation of the algorithm, $(\neg\Phi \vee \Psi)$-states in MRM $\mathcal{M}$ have been made absorbing, yielding $\mathcal{M}[\neg\Phi \vee \Psi] = ((S, \boldsymbol{R}', Label), \rho', \iota')$. This algorithm employs two matrices to store F^j and F^{j+1}.

Algorithm 4.6 Discretization

```
Discretization(InitialState s, SetOfStates $Sat_\Phi$, SetOfStates $Sat_\Psi$, TimeUpperBound
$t$, RewardUpperBound $r$, DiscretizationFactor $d$): Probability
  initialize integer $T = \frac{t}{d}$, integer $R = \frac{r}{d}$
  initialize real $Prob = 0$
  initialize matrix $F^j[|Sat_\Phi \cup Sat_\Psi|, R]$, matrix $F^{j+1}[|Sat_\Phi \cup Sat_\Psi|, R]$
  $F^j[s, \rho'(s)] = \frac{1}{d}$
  for $t = 2$ to $T$
    for each $s \in Sat_\Phi \cup Sat_\Psi$
      for $k = 1$ to $R$
        $F^{j+1}[s,k] = F^j[s, k - \rho'(s)] \cdot (1 - E'(s) \cdot d)$
                 $+ \sum_{s' \in Sat_\Phi \cup Sat_\Psi} F^j[s', k - \rho'(s') - \frac{\iota'(s',s)}{d}] \cdot \boldsymbol{R}'_{s',s} \cdot d$
      end for
    end for
    $F^j = F^{j+1}$
  end for
  for each $s \in Sat_\Psi$
    for $k = 1$ to $R$
      $Prob = Prob + F^j[s,k] \cdot d$
    end for
  end for
  return $Prob$
end Discretization
```

4.5.1 Computational Requirements

Both matrices F^j and F^{j+1} need to be stored to obtain the complete matrix F^{j+1}. A sparse representation has been used for the representation of these matrices however, in the worst case $2 \cdot |S| \cdot R$ floating point numbers need to be stored. A total of $(\frac{t}{d})$ iterations are necessary to obtain the result. The time complexity of this method is $\mathcal{O}(|S|^2 \cdot t \cdot |(t-r)| \cdot d^{-2})$.

The computational effort reported in [Tij02] is $\mathcal{O}(|S| \cdot t \cdot |(t-r)| \cdot d^{-2})$. However, in each iteration for discretized interval in time and reward every transition in MRM needs to be explored which in the worst case amounts to $|S|^2$. Consequently, the computational cost is $\mathcal{O}(|S|^2 \cdot t \cdot |(t-r)| \cdot d^{-2})$.

4.6 Uniformization Applied to Until Formulas

Uniformization can be applied to evaluate until formulas where $I = [0,t]$ and $J = [0,r]$. In this report solutions for until formulas with general time and reward bounds are not considered. Note that neither [Bai00] nor [Hav02] have considered computational methods for general time and reward bounds.

To determine the validity of formula $\mathcal{P}_{\trianglelefteq p}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$, consider the MRM $\mathcal{M} = ((S, \boldsymbol{R}, Label), \rho, \iota)$. By making certain states absorbing and by the application of theorems 4.1 and 4.3:

$$\begin{aligned} P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi) &= P^{\mathcal{M}[\neg\Phi\vee\Psi]}(s_0, \mathtt{tt}\mathcal{U}^{[t,t]}_{[0,r]}\Psi) \\ &= \Pr\{Y(t) \leq r, X(t) \vDash \Psi\} \text{ over } \mathcal{M}[\neg\Phi \vee \Psi] \text{ and } \underline{p}_{s_0}(0) = 1. \end{aligned}$$

Now it is possible to obtain the value of $P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$. To achieve this $\mathcal{M}[\neg\Phi \vee \Psi]$ is uniformized to obtain $\mathcal{M}[\neg\Phi \vee \Psi]^u = (S, \boldsymbol{P'}, \Lambda, Label, \rho', \iota')$. By discarding those finite paths in $\mathcal{M}[\neg\Phi \vee \Psi]^u$ that do not satisfy Ψ-formula in the last state and those paths whose probability is less than the truncation probability w, $P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ can be determined using equation (4.3). With initial probability distribution $\underline{p}_{s_0}(0) = 1$, $P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ can be approximated by:

$$P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi) \approx \sum_{\substack{\sigma \in TP(\mathcal{M}[\neg\Phi\vee\Psi], w, s_0, t), \\ last(\sigma) \vDash \Psi}} P(\sigma, t) \cdot \Pr\{Y(t) \leq r | \sigma\}. \tag{4.5}$$

Having found $P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$, the validity of $s_0 \vDash \mathcal{P}_{\trianglelefteq p}(\Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ may now be determined as follows:

$$s_0 \vDash \mathcal{P}_{\trianglelefteq p}(\Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi) = (P^{\mathcal{M}}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi) \trianglelefteq p).$$

In section 4.6.1 the error introduced by the truncation of paths is presented. Subsequently algorithms for the computation of the path probabilities and the conditional probability are presented in section 4.6.2 and in section 4.6.3 respectively.

4.6.1 Error Bounds

Given a path σ of length n such that $P(\sigma, t) < w$, it implies that σ and suffixes of σ are to be discarded. Let $pre(\sigma)$ be path obtained by removing the last state in σ. Amongst the paths to be discarded there is a subset of paths:

$DP(\mathcal{M}, w, s, t) = \{\sigma \in FPaths^{\mathcal{M}^u}(s) | \sigma \notin TP(\mathcal{M}, w, s, t) \wedge pre(\sigma) \in TP(\mathcal{M}, w, s, t)\}$.

The error introduced [Qur96] by discarding σ and all its suffixes is:

$$E^{\mathcal{M}[\neg\Phi\vee\Psi]^u}(\sigma, t) = P(\sigma) \cdot \left(1 - \sum_{i=0}^{n-1} \frac{e^{-\Lambda t}(\Lambda t)^i}{i!}\right).$$

The error bound for until formula computed using equation (4.5) is the error introduced by discarding all paths that are in $DP(\mathcal{M}[\neg\Phi \vee \Psi], w, s_0, t)$ and their suffixes. A slightly better bound can be obtained if further restriction is imposed that the last state in the path satisfies the formula $(\Phi \ \vee \ \Psi)$. This is since once a $(\neg\Phi \wedge \neg\Psi)$-state is reached the until formula can never be satisfied. Consequently the error bound for $\mathcal{P}_{\trianglelefteq p}(s_0, \Phi\mathcal{U}^{[0,t]}_{[0,r]}\Psi)$ and MRM $\mathcal{M}$ given w computed using equation (4.5) is:

$$E^{\mathcal{M}[\neg\Phi\vee\Psi]^u}_{\mathcal{U}}(w, t) = \sum_{\substack{\sigma \in DP(\mathcal{M}[\neg\Phi\vee\Psi], w, s_0, t) \ \wedge \\ last(\sigma) \models (\Phi \ \vee \ \Psi)}} E^{\mathcal{M}[\neg\Phi\vee\Psi]^u}(\sigma, t). \tag{4.6}$$

4.6.2 Generation of Paths and Path Probabilities

In this section a method to generate paths is presented. Using this method it is possible to generate the paths to compute the value of equation (4.5). The generation of paths can either be performed in breadth-first or depth-first manner. Of particular interest with reference to the evaluation of equation (4.5) is the depth-first strategy. This choice has two consequences. The first is that it is now easier to perform conditioning on the basis of satisfaction of formulas and path truncation. Secondly the amount of storage space required for the exploration of paths is reduced since the generation of paths of length $(N + 1)$ in the breadth-first strategy requires storage of all paths of length (N). This situation does not occur in the depth-first strategy since only a single path is stored at a time.

Consider algorithm 4.7. In the first statement, the end-condition of the path generation is specified. If a state that satisfies neither Φ nor Ψ or the probability of path is less than the truncation probability then further exploration of paths is not necessary (cf. line 1). If the present-state in a path of length n is a Ψ-state then the path satisfies the path formula and the details regarding the path are stored (cf. line 2). Subsequently, all the successor states of the present state are analyzed by a recursive call (cf. line 4). The Poisson probabilities are computed in a recursive fashion:

$$P_i = \begin{cases} \dfrac{\Lambda \cdot t}{i+1} \cdot P_{i-1} & \text{if } i > 0, \\ \\ e^{-\Lambda \cdot t} & \text{if } i = 0. \end{cases}$$

Algorithm 4.7 Depth First Path Generation

```
DFPG(integer $n$, Vector $\underline{k}$, Vector $\underline{j}$, State $s$, Probability $p$): void
  if $s \models (\neg\Phi \wedge \neg\Psi)$ or $p < w$ then return
  if $s \models \Psi$ then store$(n, \underline{k}, \underline{j}, s, p)$
  for all transitions starting from $s \in S$ to $s' \in S$
    DFPG$(n+1, \underline{k} + \underline{1}_{[s']}, \underline{j} + \underline{1}_{[s,s']}, s', \frac{\Lambda t}{n+1} \cdot p \cdot \boldsymbol{P}_{s,s'})$
  end for
end DFPG
```

$\underline{1}_{[s']}$ is a vector of length $|\underline{k}|$ and the value at index corresponding to $\rho(s')$ in $\underline{k}$ is 1 and is 0 for other indices.
$\underline{1}_{[s,s']}$: is a vector of length $|\underline{j}|$ and the value at index corresponding to $\iota(s,s')$ in $\underline{j}$ is 1 and is 0 for other indices.

Example 4.3 *Consider the WaveLAN modem example with rates as in example 4.2 and the paths to be generated for evaluating the formula $\mathcal{P}_{\trianglelefteq p}(idle\,\mathcal{U}^{[0,1]}_{[0,1000]}\,busy)$. Initially, all the $(\neg idle \vee busy)$-states are made absorbing. Subsequently, the MRM is uniformized and the algorithm DFPG is used to obtain the paths required.*

If the depth is restricted to two transitions and the starting state is state 3 then the paths generated are shown in figure 4.3. The figure also indicates the order in which the states are visited and the stored values for two paths. The values stored include the probability of the path, the vectors $\underline{k}$ and $\underline{j}$ which characterize the path. Note that in principle the truncation is based on the truncation probability w, thus the depth can be different for different paths. This is not portrayed in figure 4.3.

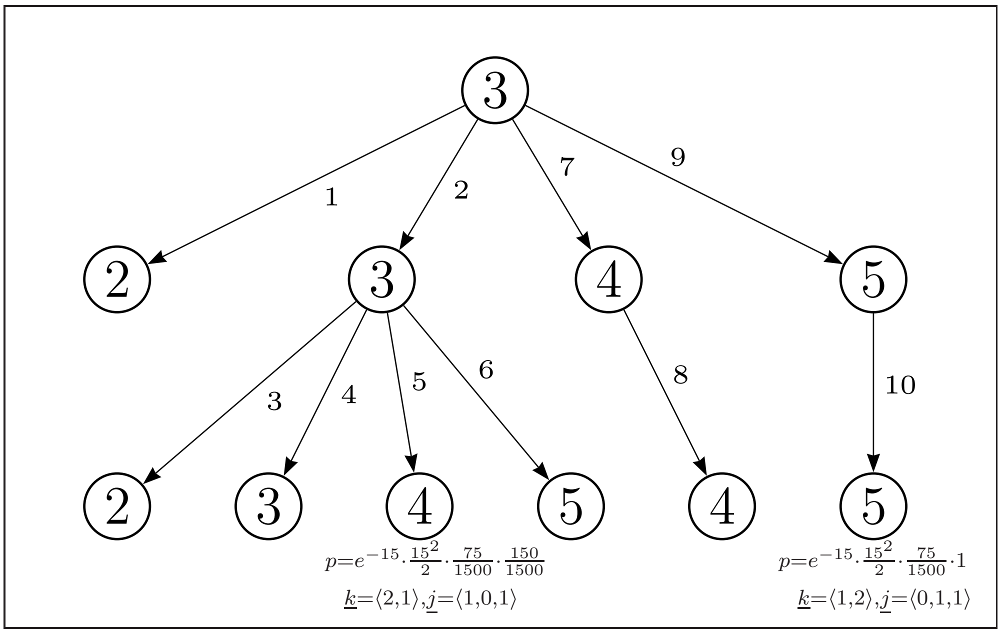

3
1
2
7
9
2
3
4
5
3
4
5
6
8
10
2
3
4
5
4
5
$p=e^{-15}\cdot\frac{15^2}{2}\cdot\frac{75}{1500}\cdot\frac{150}{1500}$
$\underline{k}=\langle 2,1\rangle, \underline{j}=\langle 1,0,1\rangle$
$p=e^{-15}\cdot\frac{15^2}{2}\cdot\frac{75}{1500}\cdot 1$
$\underline{k}=\langle 1,2\rangle, \underline{j}=\langle 0,1,1\rangle$

Figure 4.3: Depth First Path Generation

4.6.3 Computation of Conditional Probability

Methods for the computation of conditional probability given by $\Pr\{Y(t) \leq r|\sigma\}$ or $\Pr\{Y(t) \leq r|n, \underline{k}, \underline{j}\}$ are presented by de Souza e Silva & Gail [Sil94] and Qureshi & Sanders [Qur94, Qur96]. This section presents a construction similar to Qureshi & Sanders [Qur94] and new methods of computation and algorithms to compute the value of the conditional probability.

Let $U_1(t), U_2(t) \cdots, U_n(t)$ be independent, identical and uniformly distributed random variables over $(0, t)$. Now let $U_{(j)}(t)$ be the j-th smallest amongst the random variables, $U_{(0)}(t) = 0$ and $U_{(n+1)}(t) = t$. Then $U_{(1)}(t), U_{(2)}(t) \cdots, U_{(n)}(t)$ are the order statistics of $U_1(t), U_2(t) \cdots, U_n(t)$. It is known [Ros95] that the joint distribution of transition times of CTMC given n transitions of the Poisson process in $[0, t]$ is identical to the joint distribution of the order statistics of n uniformly distributed random variables over $[0, t]$.

Now let $Y_1, Y_2, \cdots, Y_{n+1}$ be the $(n + 1)$ residence times such that:

$$\begin{aligned}
Y_1 &= U_{(1)}(t) - U_{(0)}(t) = U_{(1)}(t), \\
Y_2 &= U_{(2)}(t) - U_{(1)}(t), \\
&\vdots \\
Y_{n+1} &= U_{(n+1)}(t) - U_{(n)}(t) = t - U_{(n)}(t).
\end{aligned}$$

Now let $l_1, l_2, \cdots, l_{K+1}$ be the $(K+1)$ sum of sojourn times, where l_i is the sum of lengths of intervals of state reward r_i. The *exchangeability property* [Ros01] states that random variables Y_i are exchangeable i.e.:

$$\Pr\{Y_1 \leq t_1, Y_2 \leq t_2, \cdots, Y_{n+1} \leq t_{n+1}\} = \Pr\{Y_{h_1} \leq t_1, Y_{h_2} \leq t_2, \cdots, Y_{h_{n+1}} \leq t_{n+1}\},$$

for all permutations h_i of $1, 2, \cdots, n + 1$; hence, these l_i's can now be written as:

$$\begin{aligned}
l_1 &= Y_1 + Y_2 + \cdots + Y_{k_1}, \\
l_2 &= Y_{k_1+1} + Y_{k_1+2} + \cdots + Y_{k_1+k_2}, \\
&\vdots \\
l_{K+1} &= Y_{k_1+k_2+\cdots+k_K+1} + Y_{k_1+k_2+\cdots+k_K+2} + \cdots + Y_{k_1+k_2+\cdots+k_K+k_{K+1}}.
\end{aligned}$$

Hence using the definition of Y_i:

$$\begin{aligned}
l_1 &= U_{(1)}(t) + U_{(2)}(t) - U_{(1)}(t) + \cdots + U_{(k_1)}(t) - U_{(k_1-1)}(t) = U_{(k_1)}(t), \\
l_2 &= U_{(k_1+1)}(t) - U_{(k_1)}(t) + U_{(k_1+2)}(t) - U_{(k_1+1)}(t) + \cdots + U_{(k_1+k_2)}(t) \\
&\qquad -U_{(k_1+k_2-1)}(t) = U_{(k_1+k_2)}(t) - U_{(k_1)}(t), \\
&\vdots \\
l_{K+1} &= U_{(k_1+k_2+\cdots k_K+1)}(t) - U_{(k_1+k_2+\cdots k_K)}(t) + U_{(k_1+k_2+\cdots k_K+2)}(t) - U_{(k_1+k_2+\cdots k_K+1)}(t) \\
&\qquad + \cdots + U_{(k_1+k_2+\cdots k_K+k_{K+1})}(t) - U_{(k_1+k_2+\cdots k_K+k_{K+1}-1)}(t) = t - U_{(k_1+k_2+\cdots k_K)}(t).
\end{aligned}$$

Let the state and impulse rewards be ordered: $r_1 > r_2 > \cdots > r_{K+1} \geq 0$ and $i_1 > i_2 > \cdots > i_J \geq 0$, respectively. Then the total reward accumulated $Y(t)$ given $(n, \underline{k}, \underline{j})$ is:

$$Y(t, n, \underline{k}, \underline{j}) = \sum_{i=1}^{K+1} r_i \cdot l_i + \sum_{i=1}^{J} i_i \cdot j_i. \tag{4.7}$$

Since:

$$\begin{aligned} r_1 \cdot l_1 &= r_1 \cdot U_{(k_1)}(t), \\ r_2 \cdot l_2 &= r_2 \cdot (U_{(k_1+k_2)}(t) - U_{(k_1)}(t)), \\ &\vdots \\ r_{K+1} \cdot l_{K+1} &= r_{K+1} \cdot (t - U_{(k_1+k_2+\cdots k_K)}(t)), \end{aligned}$$

hence:

$$\begin{aligned} Y(t, n, \underline{k}, \underline{j}) &= r_1 \cdot U_{(k_1)}(t) + r_2 \cdot (U_{(k_1+k_2)}(t) - U_{(k_1)}(t)) + \cdots + \\ &\qquad r_{K+1} \cdot (t - U_{(k_1+k_2+\cdots k_K)}(t)) + \sum_{i=1}^{J} i_i \cdot j_i, \\ Y(t, n, \underline{k}, \underline{j}) &= \sum_{i=1}^{K} (r_i - r_{i+1}) \cdot U_{(k_1+\cdots+k_i)}(t) + r_{K+1} \cdot t + \sum_{i=1}^{J} i_i \cdot j_i. \end{aligned}$$

The conditional probability of interest in equation (4.4) is:

$$\Pr\{Y(t) \leq r | n, \underline{k}, \underline{j}\} = \Pr\{Y(t, n, \underline{k}, \underline{j}) \leq r).$$

As $U_1(t), U_2(t), \cdots, U_n(t)$ and $t \cdot U_1(1), t \cdot U_2(1), \cdots, t \cdot U_n(1)$ have the same distribution, hence:

$$\Pr\{Y(t) \leq r | n, \underline{k}, \underline{j}\} = \Pr\{\sum_{i=1}^{K} (r_i - r_{i+1}) \cdot t \cdot U_{(k_1+\cdots+k_i)}(1) + r_{K+1} \cdot t + \sum_{i=1}^{J} i_i \cdot j_i \leq r\}, \tag{4.8}$$

which is equivalent to:

$$\Pr\{Y(t) \leq r | n, \underline{k}, \underline{j}\} = \Pr\{\sum_{i=1}^{K} (r_i - r_{i+1}) \cdot U_{(k_1+\cdots+k_i)}(1) \leq \frac{r}{t} - r_{K+1} - \frac{1}{t} \cdot \sum_{i=1}^{J} i_i \cdot j_i\}. \tag{4.9}$$

The RHS of equation (4.9) can be evaluated by the methods of Weisberg [Wei71], Matsunawa [Mat85] or Diniz, de Souza e Silva & Gail [Din02]. Previous experiments for the evaluation of the distribution of accumulated reward [Qur94, Sil94] were performed using the algorithm in [Wei71] for the computation of the distribution of a linear combination of uniform order statistics. This method however suffers from numerical instability and requires careful implementation [Qur94]. Another experiment [Qur96] using the algorithm in [Mat85] also faced difficulties in implementation.

A new and numerically stable algorithm is presented in [Din02] for the computation of equations of the form of equation (4.9) in a recursive manner. The stability of this algorithm is attributed to calculations being restricted to multiplications of real numbers in $[0, 1]$. This new algorithm additionally is simple to implement compared to the previously used methods. Consequently this algorithm is used for the computation of the distribution of the linear combination of uniform order statistics here.

Consider a linear combination of order statistics:

$$G = \sum_{i=1}^{n} a_i \cdot U_{(i)}.$$

Given that $U_{(i)} = \sum_{j=1}^{i} Y_j$ where $Y_j = U_{(j)} - U_{(j-1)}$. Then by substitution:

$$G = \sum_{i=1}^{n+1} d_i \cdot Y_i,$$

where $d_i = a_i + \cdots + a_n$.

Let $\mathcal{C} = \{c_1, c_2, \cdots, c_S\}$ be a set of distinct d_i and $\underline{k} = \langle k_1, k_2, \cdots, k_S\rangle$ where k_l is the number of intervals associated to c_l. Now:

$$\begin{aligned}\Pr\{Y(t) \le r | n, \underline{k}, \underline{j}\} &= \Pr\{\sum_{i=1}^{K}(r_i - r_{i+1}) \cdot U_{(k_1+\cdots+k_i)}(1) \le \underbrace{\frac{r}{t} - r_{K+1} - \frac{1}{t} \cdot \sum_{i=1}^{J} i_i \cdot j_i\}}_{r'} \\ &= \Omega(r', \underline{k}). \end{aligned} \quad (4.10)$$

Now define two sets: $\mathcal{G} = \{l | c_l > r'\}$ and $\mathcal{L} = \{l | c_l \le r'\}$. Given state space S, $(\underline{k}_S)_l = \{k_l \text{ if } l \in S\}$ and $\|\underline{k}_S\| = \sum_{l \in S} k_l$. Since sets $\mathcal{G}$ and $\mathcal{L}$ partition the set $\{1, 2, \cdots, S\}$, hence $\underline{k}_{\mathcal{G}} + \underline{k}_{\mathcal{L}} = \underline{k}_S$. The algorithm presented by Diniz, de Souza e Silva & Gail [Din02] is as follows:

Algorithm 4.8 Omega $\Omega(r, \underline{k})$
For $\|\underline{k}_{\mathcal{G}}\| > 0, \|\underline{k}_{\mathcal{L}}\| > 0$ choose $i \in \mathcal{G}$ and $j \in \mathcal{L}$ such that both $\underline{k}_i > 0$ and $\underline{k}_j > 0$, then:

$$\Omega(r, \underline{k}) = \left(\left(\frac{c_i - r}{c_i - c_j}\right) \cdot \Omega(r, \underline{k} - \underline{1}_j) + \left(\frac{r - c_j}{c_i - c_j}\right) \cdot \Omega(r, \underline{k} - \underline{1}_i)\right),$$

with initial conditions: for either $\|\underline{k}_{\mathcal{G}}\| = 0$ or $\|\underline{k}_{\mathcal{L}}\| = 0$ (but not both):

$$\Omega(r, \underline{k}) = \begin{cases} 1 \text{ if } \|\underline{k}_{\mathcal{G}}\| = 0, \\ 0 \text{ if } \|\underline{k}_{\mathcal{L}}\| = 0. \end{cases}$$

Recall that $\underline{1}_{[j]}$ is a vector of length $|\underline{k}|$ and the value at index corresponding to j in $\underline{k}$ is 1 and is 0 for other indices.

Example 4.4 *This hypothetical example demonstrates the dynamics of the algorithm 4.8:*

Assume:

- $(K+1)$ *distinct state rewards to be* $5 > 3 > 1 > 0$ *where* $K = 3$, *and*
- J *distinct impulse rewards to be* $2 > 1 > 0$ *where* $J = 3$.

Given a path such that $n = 6$, $\underline{k} = \langle 1,2,2,2\rangle$, $\underline{j} = \langle 4,2,0\rangle$, $t = 5$ *and* $r = 15$. $\Omega(r', \underline{k})$ *for this path can be found as demonstrated below.*

Solution: the computation of r', $\underline{a}$ *and* $\underline{d}$:

$$r' = \frac{r}{t} - r_{K+1} - \frac{1}{t} \cdot \sum\nolimits_{i=1}^{J} i_i \cdot j_i = \frac{15}{5} - 0 - \frac{1}{5} \cdot (2 \cdot 4 + 1 \cdot 2 + 0 \cdot 0) = 1,$$

$$\begin{aligned} a_1 &= r_1 - r_2 = 5 - 3 = 2, \\ a_2 &= r_2 - r_3 = 3 - 1 = 2, \\ a_3 &= r_3 - r_4 = 1 - 0 = 1, \end{aligned}$$

$$\begin{aligned} d_1 &= a_1 + a_2 + a_3 = 2 + 2 + 1 = 5, \\ d_2 &= a_2 + a_3 = 2 + 1 = 3, \\ d_3 &= a_3 = 1, \\ d_4 &= 0. \end{aligned}$$

Hence $\underline{d} = \langle 5,3,1,0\rangle$, $\underline{c} = \langle 5,3,1,0\rangle$, $\underline{k} = \langle 1,2,2,2\rangle$, *and sets* $\mathcal{G} = \{1,2\}$ *and* $\mathcal{L} = \{3,4\}$. *The recursion of* $\Omega(r', \underline{k})$ *for* $\underline{k} = \langle 1,2,2,2\rangle$ *is:*

$$\begin{tikzcd}
 & \langle 0,1,0,0\rangle \arrow[d] & \langle 0,2,0,0\rangle \arrow[d] & \langle 1,2,0,0\rangle \arrow[d] \\
\langle 0,0,0,1\rangle \arrow[r] & \langle 0,1,0,1\rangle \arrow[r] \arrow[d] & \langle 0,2,0,1\rangle \arrow[r] \arrow[d] & \langle 1,2,0,1\rangle \arrow[d] \\
\langle 0,0,0,2\rangle \arrow[r] & \langle 0,1,0,2\rangle \arrow[r] \arrow[d] & \langle 0,2,0,2\rangle \arrow[r] \arrow[d] & \langle 1,2,0,2\rangle \arrow[d] \\
\langle 0,0,1,2\rangle \arrow[r] & \langle 0,1,1,2\rangle \arrow[r] \arrow[d] & \langle 0,2,1,2\rangle \arrow[r] \arrow[d] & \langle 1,2,1,2\rangle \arrow[d] \\
\langle 0,0,2,2\rangle \arrow[r] & \langle 0,1,2,2\rangle \arrow[r] & \langle 0,2,2,2\rangle \arrow[r] & \langle 1,2,2,2\rangle
\end{tikzcd}$$

The values in each step of the above recursion are computed using the formula in the algorithm and the vector $\underline{c}$.

In conclusion for determining $P^{\mathcal{M}}(s, \texttt{tt}\,\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ given a truncation probability w initially algorithm 4.7 is used to obtain all the paths in which the last state satisfies the state formula Ψ. Each of these paths is characterized with a certain value of $\underline{k}$ and $\underline{j}$ vectors. The path probabilities of paths that are characterized with the same value of these vectors are added. Subsequently, for each path the value of the conditional probability in equation (4.5) is determined by the use of equation (4.9) and algorithm 4.8. Then $P^{\mathcal{M}}(s, \texttt{tt}\,\mathcal{U}^{[t,t]}_{[0,r]}\Psi)$ is determined using of equation (4.5) and the error bound is found using equation (4.6).

4.6.4 Computational Requirements

While the computation of equation (4.5) and of equation (4.6) depends on the nature of the underlying model; upper bounds for the computation can be provided under suitable assumptions. Since algorithm 4.8 can only start once all requisite paths are found, when state rewards and impulse rewards are present and assuming that every state is reachable in one step from every other state and all 1-step probabilities are equal; the computation of the requisite path probabilities requires $\mathcal{O}\left(\frac{N(1-N^{n-1})}{1-N}\right)$ multiplications and the computation of the until formula given the path probabilities requires $\mathcal{O}\left(\binom{K+n+1}{n+1} \times \binom{J+n}{n} \times \frac{n^2}{2}\right)$ multiplications where n is the number of steps and N is the number of states.

Although this solution is very time-complex, two problems have been resolved viz. both the computation of path probabilities and the conditional probability are done in a depth-first manner and consequently do not require substantial storage; secondly most earlier solutions were susceptible to numerical instability which is not the case with this solution.

Chapter 5

Experimental Results

This chapter presents experiments for Model Checking MRMs primarily for the computation of transient measures. An overview of the implementation of a prototype model checker is provided in section 5.1*. Section* 5.2 *describes experiments performed for ascertaining the correctness of the implementation by comparing the results to reference values. In section* 5.3*, experiments with impulse rewards by the use of both uniformization and discretization methods are presented.*

5.1 Implementation

In the context of this thesis a prototype model checker for MRMs is implemented in Java. The components of the model checker and their functions are as follows:

1. User Interface: This component functions as an interface between users and the model checker. Through this component, users input MRMs and the CSRL formulas whose validity is to be checked.

2. Model: This component stores the state space and the reward assignment functions of the MRMs. It also implements model checking functionality for until, next, steady-state, complement, and disjunction operators.

3. CSRL Parser: This component accepts a CSRL formula from Model Checker, parses the formula and then returns the parse tree.

4. Model Checker: This component invokes CSRL Parser to parse the input formula. Based on the parse tree it invokes Model for individual operators.

5. Numerical Analysis Component: This component performs several numerical analysis functions namely discretization, uniformization and Gauss-Seidel algorithm to solve linear equations and is invoked by Model.

6. Graph Analysis Component: This components performs graph analysis for finding the BSCC in MRMs and is invoked by Model.

The components and their relationship with each other are shown in figure 5.1. A usage manual for the model checker is presented in Appendix A.

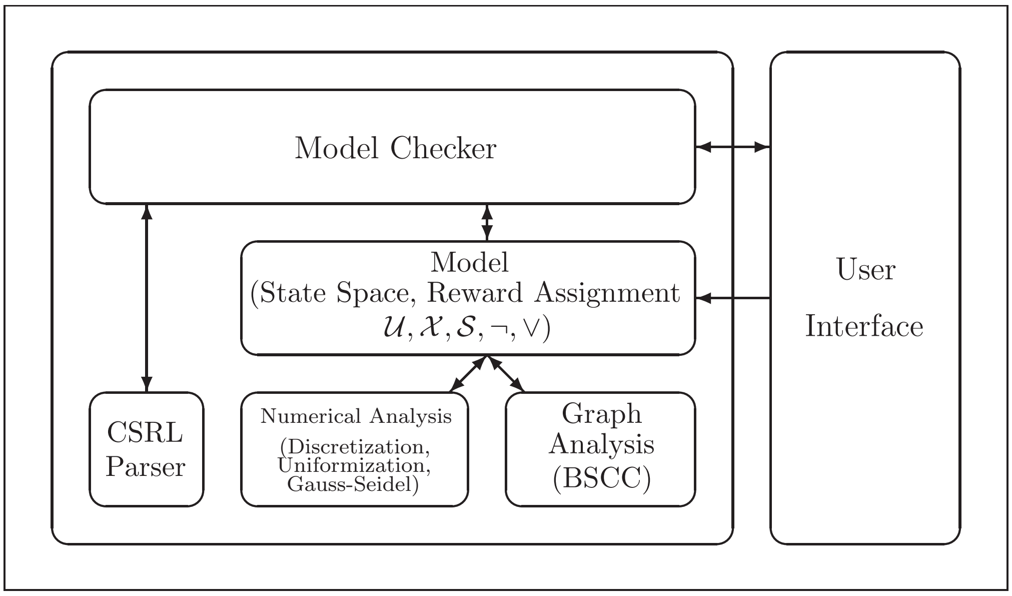


Figure 5.1: Model Checker for MRMs

5.2 Results without Impulse Rewards

To investigate the correctness of the implementation of the discretization method the case study in [Hav02] is used. In this case study only state-based reward rates are present. However, the generic algorithm for both state-based reward rates and the impulse reward also works for models with only state-based reward rates if impulse reward of zero are assigned to all transitions. In this experiment values for the until formula $\mathcal{P}_{>0.5}((Call\_Idle \vee Doze)\mathcal{U}^{\leq 24}_{\leq 600} Call\_Initiated)$ are obtained. Initially $\mathcal{M}[\neg(Call\_Idle \vee Doze) \vee Call\_Initiated]$ is obtained, which has three transient and two absorbing states. Subsequently by applying discretization method the results shown in table 5.1 with state 1 as the initial state are obtained.

Table 5.1: Result without Impulse Rewards

d	$\Pr\{Y(24) \leq 600, X(24) \models \Psi\}$	Computation Time(s)
1/16	0.49564786212263934	7.990
1/32	0.49545079878452436	65.858
1/64	0.49534976475617837	518.674

A reference value of 0.49540399 with an error bound of 10^{-8} [Hav02] is available for this computation. The values obtained by the generalized algorithm presented here converge to this reference value.

5.3 Results with Impulse Rewards

5.3.1 Experimental Setup

Experiments are performed using the model of a *triple-modular redundant* (TMR) system. The TMR system comprises of three modules performing identical tasks and a voter. The voter accumulates results from all the modules and gives a verdict only if at least two of the modules are working. If two or more modules or the voter have failed then the system is in a state of non-operation or is said to have failed. When one of the modules fails, its repair starts immediately. If the voter fails and is subsequently repaired then the system is said to start as 'new'. However, to start such repairs substantial effort is required and subsequently the repair operation consumes substantial amount of resources and consequently increases the cost of processing a single request. Such a system can be effectively modeled by means of a Markov Reward Model as in figure 5.2.

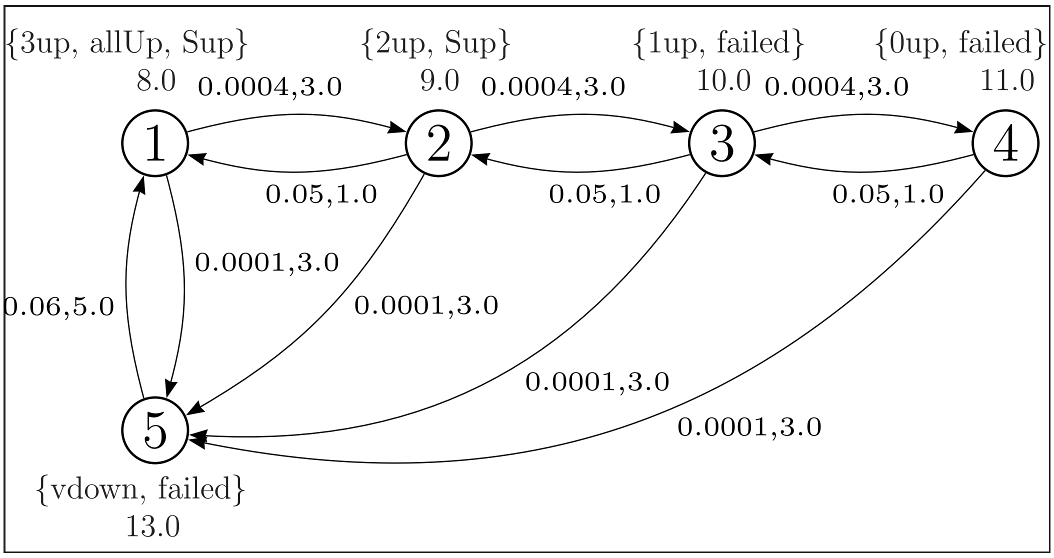


Figure 5.2: Triple-Modular Redundant System

The rates of failure and repair of components in the TMR system are specified in table 5.2. The rate of failure of components in a redundant modular system are typical (see [Tri92]). The rewards can be interpreted in two ways either as gain or loss of resources. In [Sil94] rewards are interpreted as the cost of performing repairs for faulty systems in monetary units. In the experiments conducted no explicit units are given, however, it is assumed that certain resources are being consumed during the running time of the system. When components of the system are to be repaired then it is further assumed that resources are spent at a higher rate.

The atomic propositions are interpreted as follows: *3up* means that the number of running modules is 3; *2up* means that the number of running modules is 2;

Table 5.2: Rates of the TMR Model

Transition	**Rate**
failure of modules.	0.0004 hours^{-1}.
failure of voter.	0.0001 hours^{-1}.
repair of modules.	0.05 hours^{-1}.
repair of voter.	0.06 hours^{-1}.

$1up$ means that the number of running modules is 1; $0up$ means that the number of running modules is 0; Sup means that the system is in a state of operation; $allUp$ means that all modules are running; $vdown$ means that the voter is down and $failed$ means that the system is non-operational.

All experiments are conducted on a computer with an Intel PIII processor with a clock at 868 MHz., 256 MB of RAM running Windows XP professional service pack 1. The implementation was performed in Java and executed on Java 2 runtime environment v. SE 1.4.1_02.

5.3.2 Results by Uniformization

Maintaining Constant Value for truncation probability w

In this experiment the impact of maintaining a constant truncation probability w on the probability of satisfying the given formula P, the error bounds E computed by equation (4.6) and the computation time T is investigated. The validity of the formula $\mathcal{P}_{>0.1}(Sup\ \mathcal{U}^{\leq t}_{\leq 3000}\ failed)$ for state 1 is checked with t ranging from 50 to 500. By maintaining the value of truncation probability at $w = 10^{-11}$ the results obtained are shown in table 5.3.

Table 5.3: Maintaining Constant Value for Truncation Probability

t	P	E	T (s)
50	0.005087386344177422	$2.4358698148888235 \times 10^{-9}$	0.01
100	0.010200965534212462	$1.2515341178826049 \times 10^{-8}$	0.02
150	0.015292345758962047	$3.082240323341275 \times 10^{-8}$	0.04
200	0.020357846035241836	$9.586925654419818 \times 10^{-8}$	0.08
250	0.025397296769503298	$2.23071030162702 \times 10^{-7}$	0.161
300	0.0304108011763401	$3.719970665306907 \times 10^{-7}$	0.29
350	0.035398424356873154	$8.059405465802234 \times 10^{-7}$	0.481
400	0.037778881862768586	$1.8187796388985496 \times 10^{-5}$	0.791
450	0.035702997386052426	$2.09565155821465 \times 10^{-3}$	1.142
500	0.033399142731982794	$1.19809420907302 \times 10^{-2}$	1.512

The result shows that up to $t = 350$, the error bounds are still acceptable. For values of t from 400 to 500, the error bounds grow quite fast. The increase of error bounds is since for higher values of t, the term $e^{-\Lambda t}$ becomes smaller and therefore the number of paths that are generated and computed is reduced. The graph of computation time for increasing t is shown in figure 5.3. Figure 5.3 shows that even though w is fixed, the computation time still grows fast.

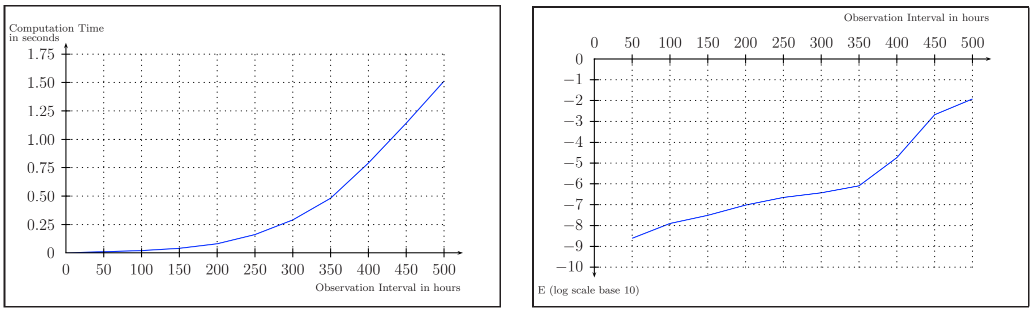


Figure 5.3: T vs. t and E vs. t for constant $w = 10^{-11}$

Maintaining Error Bound

In this experiment the impact of maintaining the error bounds E below 10^{-4} on the time of computation T is investigated. Using the same TMR system model, the validity of formula $\mathcal{P}_{>0.1}(Sup\ \mathcal{U}^{\leq t}_{\leq 3000}\ failed)$ for state 1 is checked with t ranging from 50 to 500. To maintain the magnitude of error bounds, the value of truncation probability w needs to be adjusted for each value of t. The result of the experiment is shown in table 5.4.

Table 5.4: Maintaining Error Bound

t	w	P	E	$T(s)$
50	10^{-6}	0.005066346970920541	$4.260913148296264 \times 10^{-5}$	0.00
100	10^{-7}	0.010192188416409224	$2.1869525322217564 \times 10^{-5}$	0.01
150	10^{-7}	0.01526891561598995	$5.647390585961248 \times 10^{-5}$	0.01
200	10^{-8}	0.02034951753667224	$1.810687989884388 \times 10^{-5}$	0.02
250	10^{-8}	0.02535926036855204	$6.703496676818091 \times 10^{-5}$	0.02
300	10^{-9}	0.0303887127539854	$3.0501927783531565 \times 10^{-5}$	0.07
350	10^{-10}	0.035379256114703495	$2.294785264519215 \times 10^{-5}$	0.21
400	10^{-11}	0.037778881862768586	$1.8187796388985496 \times 10^{-5}$	0.791
450	10^{-12}	0.03777910398006526	$1.743339250561631 \times 10^{-5}$	2.373
500	10^{-13}	0.037779567600526885	$1.6531714588135478 \times 10^{-5}$	8.762

The last column of table 5.4 shows the computation time for increasing t. This computation time grows significantly faster when the error bound is to be maintained than the case where it was not to be maintained. This is due to longer paths being explored as the value of the truncation probability is lowered. As the number of paths to be explored grows exponentially in the worst case with the depth of exploration; the computation time grows faster when the error bound is to be maintained than the case where it was not to be maintained.

The Ability of the System to Reach the Fully Operational State with Constant Failure Rate of Modules

This section presents the results for an alternative until formula. The system has 11 identical modules and a voter. A constant rate of failure and repair of the modules and the voter is considered with rates as in previous sections. This alternative formula concerns the ability of the system to reach the fully operational state i.e. all the modules are working, starting from a state representing a certain number of working modules. It is formulated in CSRL as follows:

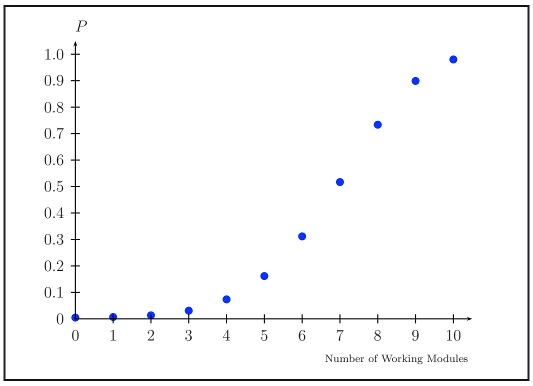
$$\mathcal{P}_{>0.1}(\texttt{tt}\ \mathcal{U}^{\leq 100}_{\leq 2000} allUp)$$

where $t = 100$ and $r = 2000$. This implies that given a starting state the probability of reaching the fully operational state within 100 hours and by expending resources less than 2000 units is investigated. A truncation probability $w = 10^{-8}$ is used. Table 5.5 presents the results of this experiment. The number of working modules in the starting state is represented by n.

Table 5.5: Reaching the Fully Operational State with Constant Failure Rates

n	P	E	T (s)
0	0.00482952588914756	$4.05866323902596 \times 10^{-4}$	0.381
1	0.0068486521925764	$4.19455701443569 \times 10^{-4}$	0.481
2	0.0131488893307554	$3.82813317721167 \times 10^{-4}$	0.42
3	0.0307864803541378	$3.01314786268715 \times 10^{-4}$	0.401
4	0.0735906999244802	$2.44049258515375 \times 10^{-4}$	0.35
5	0.161653274832831	$1.66495488214506 \times 10^{-4}$	0.261
6	0.311639369763902	$1.20696967385326 \times 10^{-4}$	0.23
7	0.516966415983422	$7.02115774733882 \times 10^{-5}$	0.11
8	0.733673548795558	$3.47684889215192 \times 10^{-5}$	0.06
9	0.899015328912742	$1.64366888658804 \times 10^{-5}$	0.03
10	0.980329681725223	$4.57035775880327 \times 10^{-6}$	0.01

Figure 5.4 presents the results obtained for this formula. For states representing more number working modules the probability of reaching the fully operational state is much higher than for those states that represent a lesser number of working modules and the computation time is lower since lesser and more probable paths need to be explored to reach the fully operational state.



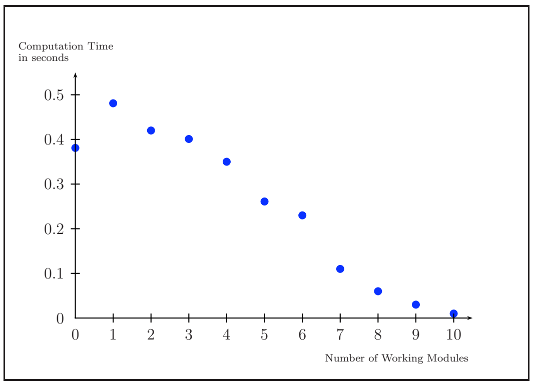


Figure 5.4: P and T vs. Number of working modules with constant failure rates

The Ability of the System to Reach the Fully Operational State with Variable Failure Rate of Modules

This section presents the results for the formula which was considered in the previous section with variable rate of failure of modules. The rates of failure and repair are in table 5.6. The formula in CSRL is as follows:

$$\mathcal{P}_{>0.1}(\texttt{tt}\,\mathcal{U}^{\leq 100}_{\leq 2000} allUp).$$

A truncation probability of 10^{-8} is used. Table 5.7 presents the results.

Table 5.6: Variable Rates

Transition	**Rate**
failure of modules.	$n \times 0.0004$ hours^{-1}.
failure of voter.	0.0001 hours^{-1}.
repair of modules.	0.05 hours^{-1}.
repair of voter.	0.06 hours^{-1}.

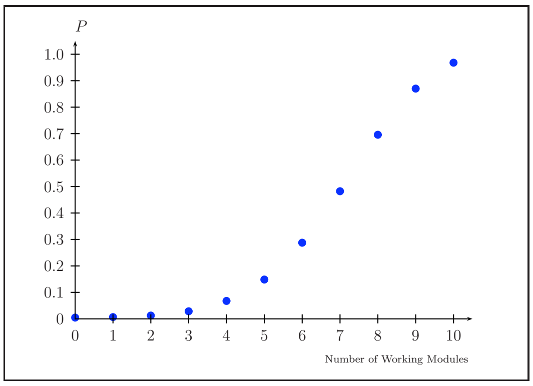
Figure 5.5 presents the results obtained for this formula. As the rate of failure of modules is higher than in the case with constant failure rate it can be observed that the probability of the system reaching the fully operational state in this case is lower than in the case with constant rates of failure.

5.3.3 Results by Discretization

In this experiment the method based on discretization is used to verify the validity of formula $\mathcal{P}_{>0.1}(Sup\,\mathcal{U}^{\leq t}_{\leq 3000}\ failed)$ for state 1 with t ranging from 50 to 200. A constant value of discretization factor $d = 0.25$ is used.The result of the experiment is shown in table 5.8.

Table 5.7: Reaching the Fully Operational State with Variable Failure Rates

n	P	E	$T\ (s)$
0	0.00477909028870443	$6.38697324029973 \times 10^{-4}$	0.49
1	0.00664628290706118	$7.20798178315112 \times 10^{-4}$	0.571
2	0.0124264528171119	$7.33708127644168 \times 10^{-4}$	0.621
3	0.0285473649414625	$7.07105529376643 \times 10^{-4}$	0.62
4	0.0676727123697789	$6.27622240550083 \times 10^{-4}$	0.611
5	0.14851270909792	$5.35659168600983 \times 10^{-4}$	0.521
6	0.287706855662473	$4.10240541832982 \times 10^{-4}$	0.4
7	0.482315748557532	$2.99067173956765 \times 10^{-4}$	0.3
8	0.695701644333058	$1.78056305155566 \times 10^{-4}$	0.18
9	0.87014207211784	$9.35552614283647 \times 10^{-5}$	0.091
10	0.968076165457539	$3.27905198638695 \times 10^{-5}$	0.04



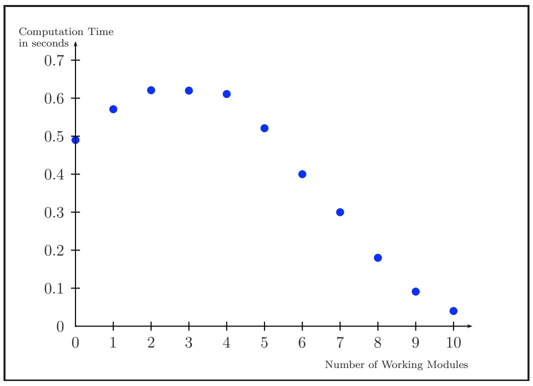


Figure 5.5: P and T vs. Number of working modules with variable failure rates

Table 5.8: Results by Discretization

t	P	$T(s)$
50	0.005061779415718182	14.409
100	0.010175568967901463	88.118
150	0.015267158582408371	345.652
200	0.020332872743413364	1592.433

It can be observed from table 5.4 and 5.8 that the results obtained using uniformization and discretization methods converge to the same value. The values obtained using the modified discretization method for the model in [Hav02] converge to reference values.

Chapter 6

Conclusions

In this thesis MRMs, the logic CSRL and its characterization with state-based reward rate have been extended to incorporate impulse rewards. Algorithms for model checking Markov Reward Models with impulse rewards and to compute error bounds for transient measures have been developed. The algorithms developed are simple to implement and are numerically stable. A prototype model checker has been implemented. An example application to demonstrate the applicability of model checking MRMs has been developed. The correctness of the implementation has been demonstrated by empirical comparison to reference values in [Hav02] when impulse rewards are not present. The correctness of the implementation when impulse rewards are present has been demonstrated by equivalence of values obtained by the use of two different methods viz. uniformization and discretization for transient measures.

The computational requirements of the algorithms for evaluating transient measures for until operator are particularly extreme. The following are some observations regarding these algorithms:

- The uniformization method is applicable when the value of (Λt) is small. As the value of (Λt) is increased the number of paths to be explored grows exponentially and makes this method unpractical.
- The discretization method is decidable in polynomial time although it too depends on the value of (Λt) where larger values of (Λt) require smaller value of discretization factor. However, the discretization method also depends on the value of the reward bound.

A further shortcoming is that methods to compute values for transient measures are restricted to reward intervals of $[0, r]$ and mission time intervals of $[0, t]$.

Future work in the direction of Model Checking Markov Reward Models includes developing efficient algorithms for computing transient measures. Another direction is to develop methods to compute values for transient measures that are not restricted to reward intervals of $[0, r]$ and mission time intervals of $[0, t]$.

Bibliography

[Bai00] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. *On the Logical Characterisation of Performability Properties.* Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, LNCS Vol. 1853, Springer-Verlag, pp. 780-791, 2000.

[Bai02] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. *Automated Performance and Dependability Evaluation using Model Checking.* Computer Performance Evaluation, LNCS Vol. 2459, Springer, pp. 261-289, 2002.

[Bai03] C. Baier, B.R. Haverkort and H. Hermanns and J.-P. Katoen. *Model-Checking Algorithms for Continuous-Time Markov Chains.* IEEE Transactions on Software Engineering, Vol. 29, Issue: 6, pp. 524-541, 2003.

[Cla99] E.M. Clarke, O. Grumberg and D. Peled. *Model Checking.* MIT Press, 1999.

[Din02] M.C. Diniz, E. de Souza e Silva and H.R. Gail. *Calculating the Distribution of a Linear Combination of Uniform Order Statistics.* INFORMS Journal on Computing, Vol. 14 No. 2, pp. 124-131, 2002.

[Hav98] B.R. Haverkort. *Performance Evaluation of Computer Communication Systems.* John Wiley & Sons, 1998.

[Hav02] B.R. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen and C. Baier. *Model Checking Performability Properties.* Dependable Systems and Networks (DSN), IEEE CS Press, pp. 103-112, 2002.

[How71] R.A. Howard. *Dynamic Probabilistic Systems Vol I. II.* John Wiley & Sons, 1971.

[Kat02] J.-P. Katoen. *System Validation - 214012(Course Notes).* University of Twente, 2002.

[Mat85] T. Matsunawa. *The Exact and Approximate Distributions of Linear Combinations of Selected Order Statistics from Uniform Distributions.* The Annals of the Institute of Statistical Mathematics, Vol. 37, pp. 1-16, 1985.

[Mey80] J.F. Meyer. *On Evaluating Performability of Degradable Computing Systems.* IEEE Transactions on Computers, Vol. C-29, pp. 720-731, 1980.

[Mey95] J.F. Meyer. *Performability Evaluation: Where It is and What Lies Ahead.* Proc. 1995 IEEE Int'l Computer Performance and Dependability Symposium, IEEE CS Press, pp. 334-343, 1995.

[Nuu93] E. Nuutila and E. Soisalon-Soininen. *On Finding the Strongly Connected Components in a Directed Graph.* Information Processing Letters, Vol. 49, pp. 9-14, 1993.

[Pau01] P.J.M. Havinga and G.J.M. Smit. *Energy-Efficient Wireless Networking for Multimedia Applications.* Wireless Communications and Mobile Computing, Vol. 1, Wiley, pp. 165-184, 2001.

[Qur94] M.A. Qureshi and W.H. Sanders. *Reward Model Solution Methods with Impulse and Rate Rewards: An Algorithm and Numerical Results.* Performance Evaluation, Vol. 20 No. 4, pp. 413-436, 1994.

[Qur96] M.A. Qureshi and W.H. Sanders. *A New Methodology for Calculating Distributions of Reward Accumulated during a Finite Interval.* Symposium on Fault-Tolerant Computing, IEEE CS Press, pp. 116-125, 1996.

[Ros95] S.M. Ross. *Stochastic Processes.* John Wiley & Sons, 1995.

[Ros01] S.M. Ross. *A First Course in Probability.* 6th edition, Prentice Hall, 2001.

[Sil94] E. de Souza e Silva and H.R. Gail. *An Algorithm to Calculate Transient Distributions of Cumulative Reward.* Tech. Rep. CSD940021, UCLA, 1994.

[Tij02] H.C. Tijms and R. Veldman. *A Fast Algorithm for the Transient Reward Distribution in Continuous-Time Markov Chains.* Operations Research Letters, Vol. 26, pp. 155-158, 2000.

[Tri92] K.S. Trivedi, J.K. Muppala, S.P. Woolet and B.R. Haverkort. *Composite Performance and Dependability Analysis.* Performance Evaluation, Vol. 14, pp. 197-215, 1992.

[Wei71] H. Weisberg. *The Distribution of Linear Combinations of Order Statistics from the Uniform Distribution.* Annals of Institute of Statistics, Vol. 42 No. 2, pp. 704-709, 1971.

Appendix: Usage Manual

A tool for model checking MRMs has been implemented in the context of this thesis. Given a Continuous-Time Markov Reward Model by providing its transition information (*.tra file*), its label information (*.lab file*) and its reward information (*.rewr* file - state rewards, *.rewi* file - impulse rewards), the tool checks the validity of properties expressed in Continuous Stochastic Reward Logic (CSRL). The tool can be used to validate all CSRL formulae. For until formulas the time and reward bounds that are supported are $[0, t]$ and $[0, r]$.

In this tool, checking the properties of a model involves finding all states that satisfy a given formula. The formula can be input such that the operators are expressed as follows:

1. True: `TT`
2. False: `FF`
3. And operator: `&&`
4. Or operator: `||`
5. Not operator: `!`
6. Infinity: `~`
7. `S(op fl) f`
8. `P(op fl) [X[fl,fl] [fl,fl] f]`
9. `P(op fl) [f U[int,int][fl,fl] f]`

where `op` is binary comparison operator, `f` is formula, `int` is integer and `fl` is float. Thus if one wishes to input a formula that states that "a b-state can be reached with probability at least 0.3 by at most 3 time-units along a-state accumulating costs at most 23" then the input will be: `P(>=0.3) [a U [0,3][0,23] b]`.

To invoke the model checker the first step is to create a Continuous-Time Markov Reward Model. The tool does not provide facilities to create such models. However, the model can be speficied conveniently using four files by providing its transition information (.tra file), its label information (.lab file) and its reward information (.rewr file - state rewards, .rewi file - impulse rewards). The format of these files is as follows:

1. Transition information (.tra file):
```
STATES n
TRANSITIONS m
state1 state2 rate
...
```
2. Label information (.lab file):
```
#DECLARATION
ap ap ...
#END
state ap[,ap]*
...
```
3. State reward information (.rewr file):
```
state reward
...
```
4. Impulse reward information(.rewi file):
```
TRANSITIONS n
state1 state2 reward
...
```

Subsequently the tool can be invoked using the following command:

```
java checker/MRMChecker *.tra *.lab *.rewr *.rewi [{u|d} = f] [NP]
```

1. `*.tra`: specify the location and name of the **.tra* file in a relative or absolute manner.

2. `*.lab`: specify the location and name of the **.lab* file in a relative or absolute manner.

3. `*.rewr`: specify the location and name of the **.rewr* file in a relative or absolute manner.

4. `*.rewi`: specify the location and name of the **.rewi* file in a relative or absolute manner.

5. `[{u|d} = f]`: specify whether uniformization or discretization should be used for until formulas. For uniformization a truncation probability w must be specified. For discretization a discretization factor d must be specified. This is optional and if nothing is specified then uniformization is used with a truncation probability (w) of 10^{-8}.

6. `[NP]`: This is optional. If specified then it indicates that the computed probabilities will not be printed; only the set of states that satisfy the formula will be printed.