Model Checking Structured Infinite Markov Chains

Anne Remke

CTIT Ph.D.-thesis Series No. 08-118
Centre for Telematics and Information Technology
University of Twente, P.O. Box 217, NL-7500 AE Enschede

ISSN 1381-3617
ISBN 978-90-8570-304-4

Publisher: Woermann Printing Service.
Cover design: Tiemen Harms.

MODEL CHECKING STRUCTURED INFINITE MARKOV CHAINS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 20 june 2008 om 15.00 uur

door

Anne Remke

geboren op 24 april 1980
te Münster, Duitsland

Dit proefschrift is goedgekeurd door
de promotor, prof. dr. ir. B.R. Haverkort.

In loving memory of my grandmother
Paula Remke

Abstract

In the past probabilistic model checking hast mostly been restricted to finite state models. This thesis explores the possibilities of model checking with continuous stochastic logic (CSL) on infinite-state Markov chains. We present an in-depth treatment of model checking algorithms for two special classes of infinite-state CTMCs: (i) Quasi-birth-death processes (QBDs) are a special class of infinite-state CTMCs that combines a large degree of modeling expressiveness with efficient solution methods. (ii) Jackson queuing networks (JQNs) are a very general class of queueing networks that find their application in a variety of settings. The state space of the CTMC that underlies a JQN, is highly structured, however, of infinite size in as many dimensions as there are queues, whereas the underlying state-space of a QBD can be seen as infinite in one dimension.

Using a new property-driven independency concept that is adapted to QBDs and JQNs, accordingly, we provide model checking algorithms for all the CSL operators. Special emphasis is given to the time-bounded until operator for which we present a new and efficient computational procedure named uniformization with representatives. By the use of an application-driven dynamic termination criterion, the algorithms stop whenever the property to be checked can be certified (or falsified).

Next to the above methodological contributions of this thesis, we also use the new techniques for an extended case study on bottlenecks in wireless two-hop ad hoc networks. The results of our analysis are compared with extensive simulations and show excellent agreement for throughput, mean number of active sources and mean buffer occupancy at the bottleneck station.

Contents

Chapter 1

Introduction

As computer- and communication systems keep growing rapidly, it is very important to be able to analyze the performance of such systems before they are actually built. It is possible to analyze computer- and communication systems in a quantitative way by using model-based performance evaluation. A variety of techniques and tools have been developed to accommodate such evaluations, e.g., based on queueing networks [25], stochastic Petri nets [4] and process algebras [44, 42].

First, a model of the real system has to be built. In the simplest case, the model reflects all possible states that the system can reach and all possible transitions between states. Continuous-time **Markov chains** (CTMCs) have been used widely for modeling and performance and dependability evaluation of computer and communication systems. CTMCs are well understood, mathematically attractive while at the same time flexible enough to model complex systems. In practice, communication systems usually need large buffer capacities, or even unbounded buffers. Analyzing a structured CTMC with infinite-state space is, in some cases, easier than analyzing a huge finite CTMC. Once the CTMC has been constructed, it is possible to calculate some performance measures of the system with a number of well-known numerical methods. Such performance measures are for example the utilization of the server, the time a customer has to spend waiting in the line or the length of the queue. For both, finite and infinite Markov chains, solution methods exist to calculate the probabilities of residing in each single state.

Model-based performance evaluation is a method to analyze the system in a quantitative way. **Model checking**, however, traditionally focuses on the qualitative evaluation of the model. As formal verification method, model checking analyzes the functionality of the system model. A property that needs to be analyzed has to be specified in a logic with consistent syntax and semantics. For every state of the model, it is then checked whether the property is valid or not. The Continuous Stochastic Logic (CSL) [8] has been introduced to express quantitative properties on finite CTMCs. Efficient computational algorithms have been developed for checking finite CTMCs against formally specified properties expressed in these logic, cf. [7, 8], as well as supporting tools, cf. PRISM [53], and ETMC^2 [43], the APNN toolbox [18], and recently MRMC [48]. Other tools, like GreatSPN [27] are used as front-end to model checking tools like PRISM and MRMC. So far, the work on model checking continuous-time Markov chains has focused on *finite*-state models. However, there are many applications for which **infinite-state models** are more appropriate: think of modeling systems with unbounded buffers, of models including variables, or of approximating the behavior of very large-but-finite systems. Model checking all CSL properties on *general* infinite-state CTMCs is, however, beyond reach.

Therefore, in this thesis we present new stochastic model checking algorithms for two classes of **structured infinite-state Markov chains**. Next to the methodological contribution of this theses, we also use the new techniques for an extended case study on bottlenecks in wireless two-hop ad hoc networks. We will address these issues in more detail below.

Quasi-birth-death models

Quasi-birth-death models (QBDs) [61] comprise a very versatile yet well-understood class of infinite-state CTMCs. It is not necessary to specify QBDs manually at the state level, as high-level specifications, like, e.g., *infinite stochastic Petri nets*, do exist [63].

In this thesis, we provide a complete description of CSL model checking algorithms for QBDs, extending [65]. We show that the syntax and semantics of CSL as for the finite case apply here as well. To facilitate the model checking algorithms, we introduce a new independency concept for CSL formulas. For model checking two of the most important CSL operators, that is, the steady-state and the probabilistic path operator of, we have to develop new algorithms. For the steady-state operator, we have to compute steady-state probabilities for QBDs; we can resort to well-known algorithms for that purpose. However, for model checking the time-bounded until operator of CSL, we also need efficient algorithms for the transient analysis of QBDs. This can be done with a new and efficient uniformization-based method, called *uniformization with representatives* which is presented in the context of model checking. The feasibility of our approach is shown in a small case study on connection management.

Jackson queueing networks

Queueing networks have been used for about half a century now, for modeling and analyzing a wide variety of phenomena in computer, communication, and logistics systems. Seminal work on queueing networks was done by Jackson in the 1950s [46, 47] in which he developed an important theorem that characterizes the steady-state probabilities to reside in certain states in a restricted class of queueing networks.

In this thesis we develop a new CSL model checking procedure for the CTMCs that underlie Jackson queueing networks (JQNs). As for CSL model checking of CTMCs, we need to be able to compute both steady-state and transient state probabilities for all states, and for all possible starting states. The key issue lies in the fact that the CTMC underlying a JQN is infinite in as many dimensions as there are queues in the JQN. For the steady-state probabilities we can rely on the seminal work of Jackson [46, 47], however, for the transient state probabilities, no results are readily available. Similar to the approach taken for QBDs, we use a uniformization-based approach to compute the transient state probabilities in JQNs that are needed to verify the validity of CSL properties. The highly structured state space allows us to conclude the validity of CSL properties for groups of states on the basis of the validity for a so-called representative state in such a group. This reduces the infinite number of state probabilities to be computed to a finite number. A small case study on an e-business site shows the feasibility or our approach.

IEEE 802.11e case study

Based on the algorithms derived for CSL model checking QBDs, we pursue an extended case study on the analysis of bottlenecks in IEEE 802.11e two-hop ad hoc networks.

In such ad hoc networks, stations that are in reach of each other all contend for the same resource, i.e., the shared ether as transmission medium. Research has shown that, effectively, the transmission medium is equally shared among contending stations [14, 58]. This leads to undesirable situations in case one of the nodes happens to function as a bridge toward either another group of nodes, or to the fixed internet, as visualized in Figure 1.1.

Recently, a quality-of-service (QoS) extension of the IEEE 802.11 standard has been proposed. We present a versatile and accurate performance model to study how these new QoS extensions can be used to improve the performance of wireless nodes competing for the transmission medium in a two-hop ad hoc network.

We use the new model checking algorithms for QBDs to evaluate this model. The results of our analysis are compared with extensive simulations (using Opnet), and show excellent agreement for throughput, mean number of active sources and mean buffer occupancy at the bottleneck station. An important asset of our model and analysis technique is that it allows for very quick evaluations: where simulations require up to one hour per scenario, our model is solved in seconds. Due to the speed and the accuracy of our analysis we are able to find those parameter settings that results in the maximum throughputs.

Outline of the thesis

This thesis provides CSL model checking algorithms for two classes of well-structured Markov chains. Furthermore we show the versatility of this approach with a detailed case study.

Chapter 2 addresses CTMCs with both finite and infinite-state space. The logic CSL is presented as a formalism to specify complex properties on states and paths of CTMCs. Furthermore, we recapitulate how the next and the until operator are model checked on finite CTMCs.

Chapter 3 introduces QBDs and addresses in detail the model checking for all CSL operators. We present an efficient uniformization-based approach to compute all required transient state probabilities. Based on this, we derive model checking algorithms for the time-bounded until, the interval until and the point interval until for infinite CTMCs of QBD type. A small case study shows the feasibility of our approach.

In **Chapter 4** we first introduce JQNs and their underlying state space, before we present an approach to structurally decompose the infinite state space that allows us to deal with it efficiently. Model checking all CSL operators is discussed, before we present a uniformization-based approach to compute the transient state probabilities, similar to the approach for QBDs. Again, this allows us to develop efficient model checking algorithms for the different flavors of the until operator. Finally, the scalability of an e-business site, modeled as JQN, is analyzed with the presented model checking techniques.

In **Chapter 5** we analyze for which extensions of QBDs, model checking with the framework proposed in Chapter 3 is still feasible. Also for JQNs several extensions and the model checking thereof are discussed. Furthermore, detailed links to related work on transient analysis of infinite CTMCs and on model checking infinite Markov

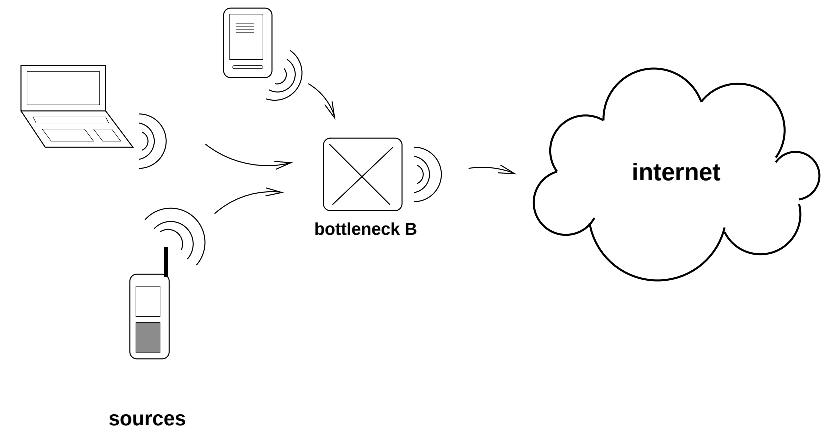


Figure 1.1: Bottleneck in a two-hop ad hoc network

chains are presented.

Then, in **Chapter 6**, we provide an elaborate case study on the analysis of bottleneck situations in IEEE 802.11e two-hop ad hoc networks, validated by detailed simulation studies performed with Opnet. The complete IEEE 802.11e access mechanism, including the QoS parameters, is addressed. We present a hierarchical modeling approach in detail and discuss the maximum throughput that can be obtained for a given set of QoS parameters.

In **Chapter 7** we summarize the contents of this thesis, and explicitly state the contributions of this thesis.

Chapter 2

Foundations

In this chapter we introduce the foundations of model checking continuous-time Markov chains (CTMCs) with continuous stochastic logic (CSL). In Section 2.1 we present the class of continuous-time Markov chains with both finite and infinite state space. Paths on CTMCs and their cylinder set are discussed in Section 2.2, before we introduce two different types of state probabilities for CTMCs in Section 2.3. The logic CSL is presented as a formalism to specify complex properties on states and paths of CTMCs in Section 2.4. Section 2.5 summarizes how the next operator, the time bounded until, the interval until and the point interval until are model checked on finite CTMCs. In Section 2.6 we discuss the general model checking routine via satisfaction sets, before we conclude in Section 2.7.

2.1 Continuous-time Markov chains

A continuous-time Markov chain (CTMC) is a stochastic process, characterized by a discrete state space $\mathsf{S} = \{0, 1, \ldots\}$, the continuous time domain $\mathcal{T} = [0, \infty)$ and the *Markov property*. This property states that the probability to reside in a given state in the near future only depends on the current state and not on the states visited before, and neither on the already passed residence time in the current state. We first present the definition of a labeled CTMC before we discuss its properties.

Definition 1 (CTMC). A labeled CTMC $\mathcal{M}$ is a tuple $(\mathsf{S}, \mathbf{T}, L)$ consisting of a countable set of states S, a transition rate matrix $\mathbf{T} : \mathsf{S} \times \mathsf{S} \Rightarrow \mathbb{R}_{\geq 0}$ and a labeling function $L : \mathsf{S} \to 2^{AP}$ that assigns atomic propositions from a fixed finite set AP to each state.

The value $\mathbf{T}(\mathsf{s}, \mathsf{s}')$, equals the rate at which the CTMC moves from a state s to state s' in one step. $\square$

Based on the transition rate matrix it is possible to express a number of other means to describe the behavior of the CTMC. The *total rate* at which any transition outgoing from state s is taken, is denoted

$$\mathbf{E}(\mathsf{s}) = \sum_{\mathsf{s}\in\mathsf{S},\ \mathsf{s}\neq\mathsf{s}'} \mathbf{T}(\mathsf{s},\mathsf{s}'). \tag{2.1}$$

A CTMC as defined above is a stochastic process $\{X(t)|t \in \mathcal{T}\}$, where $X(t) \in \mathsf{s}$ is random variable that gives the state occupied in the process at time t. For non-negative $t_0 < t_1 < \ldots < t_{n+1}$ and $x_0, x_1, \ldots, x_{n+1}$, the Markov property for a CTMC can be stated as [79]:

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

Furthermore, we require a CTMC to be time homogeneous, that is, invariant to time-shifts (with $t, s \in \mathcal{T}, t > s$):

$$\Pr\{X(t) = x \mid X(s) = x_s\} = \Pr\{X(t-s) = x \mid x(0) = x_s\}.$$

In a CTMC, the state residence times must be exponentially distributed; this is a result of the required memorylessness. The probability to leave state s before time t is exponentially distributed:

$$\Pr\{\text{leave } \mathsf{s} \text{ before } t\} = 1 - e^{-\mathbf{E}(\mathsf{s})\cdot t}.$$

The embedded discrete-time Markov chain corresponding to the CTMC is denoted as

$$\mathbf{N}(\mathsf{s},\mathsf{s}') = \frac{\mathbf{T}(\mathsf{s},\mathsf{s}')}{\mathbf{E}(\mathsf{s})} \tag{2.2}$$

and expresses the probability that the CTMC moves from state s to state s' in the next step.

The rate matrix $\mathbf{T}$ allows for self loops in the CTMC. This can be useful, because it is possible for a CTMC derived from a high-level specification to contain self loops. For performance measures and most algorithms presented in this theses, self loops do not matter as residence times in a CTMC obey a memoryless distribution, hence self loops can be eliminated. However, if only one-step probabilities are analyzed, self loops can make a difference.

When self loops are removed from the transition matrix, we obtain a square generator matrix $\mathbf{Q} : \mathsf{S} \times \mathsf{S} \to \mathbb{R}_{\geq 0}$, defined by

$$\mathbf{Q} = \mathbf{T} - \mathbf{E}, \text{ with } \mathbf{E}(\mathsf{s},\mathsf{s}') = \begin{cases} \mathbf{E}(\mathsf{s}), & \text{for } \mathsf{s} = \mathsf{s}', \\ 0, & \text{otherwise.} \end{cases}.$$

The value $\mathbf{Q}(\mathsf{s},\mathsf{s}')$, for $\mathsf{s} \neq \mathsf{s}'$, equals the rate at which a transition from state s to state s' occurs in the CTMC, whereas $\mathbf{Q}(\mathsf{s},\mathsf{s}')$ denotes the negative sum of the off-diagonal entries in the same row of $\mathbf{Q}$; its value represents the rate of leaving state s (in the sense of an exponentially distributed residence time).

Definition 2 (Irreducibility). A CTMC is called irreducible if for any two states $\mathsf{s}, \mathsf{s}' \in \mathsf{S}$, there exists $n \in \mathbb{N}$ such that $\mathbf{T}^n(\mathsf{s}, \mathsf{s}') > 0$. □

The CTMC may contain states that cannot be left anymore. In this case all outgoing rates from this state equal zero. The state is then called absorbing:

Definition 3 (Absorbing state). A state s of a CTMC is called absorbing if $\mathbf{T}(\mathsf{s}, \mathsf{s}') = 0,\ \forall \mathsf{s}' \in \mathsf{S}$. □

In order to describe the evolution of a CTMC in time completely, we need initial probabilities for the individual states. These are given by way of the initial distribution, that assigns an initial probability to every state.

Definition 4 (Initial distribution). An initial distribution on $\mathcal{M} = (\mathsf{S}, \mathbf{T}, L)$ is a function $\alpha : \mathsf{S} \to [0, 1]$ such that
$\sum_{\mathsf{s} \in \mathsf{S}} \alpha(\mathsf{s}) = 1$. □

Definition 5 (Recurrence). A state s is said to be *recurrent* if the probability to return to that state is one. A recurrent state s is said to be positive recurrent if the mean time between two successive visits to state s is finite. □

Definition 6 (Ergodicity). A state $\mathsf{s} \in \mathsf{S}$ is called *ergodic* if it is positive recurrent and aperiodic. A CTMC is called *ergodic*, if and only if the state space consists of one irreducible set of ergodic states where from every state $\mathsf{i} \in \mathsf{S}$ every other state $\mathsf{j} \in \mathsf{S}$ can be reached with a positive probability within a finite number of steps. □

Note that CTMCs are aperiodic by definition. CTMCs can either have a finite or an infinite countable state space s. The latter can be used to model systems with infinite server capacity, as for example the infinite-server queue, or models with infinite buffer. An important difference between finite and infinite-state CTMCs is that the corresponding transition rate matrices of infinite CTMCs are of infinite size.

In this thesis we deal with infinite CTMCs that exhibit a special structure. One of the simplest infinite CTMC with a special structure is the so-called birth-death process with constant rates, where from a state i for $\mathsf{i} > 0$ only transitions to the neighbors $\mathsf{i} - 1$ and $\mathsf{i} + 1$ are allowed, as illustrated in Figure 2.1, where states are depicted as nodes and transitions as arrows.

The state space $\mathsf{S} = \{0, 1, \ldots\}$ is of infinite size and can be used, for example, to represent the number of customers in a queue with negative exponentially distributed inter-arrival and service times, the so-called $M|M|1$ queue [23]. In the following, we deal with two main classes of infinite-state CTMCs, that can both be seen as an extension of the birth-death process with constant rates.

- We address *Quasi Birth Death* (QBDs) processes in Chapter 3. In a QBD, we have neighboring *levels* that consist of a group of finitely many states, instead

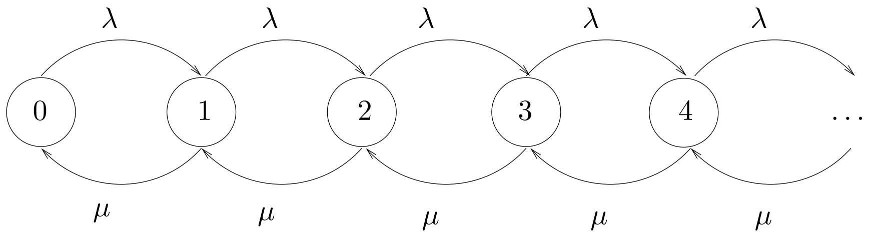


Figure 2.1: Birth-death process with constant rates

of just a single state. All levels are alike, except for the first one, that can be different. Similar to a birth-death process, in a QBD, transitions may take place between neighboring levels and within a level.

- We address the class of *Jackson queueing networks* (JQNs) in Chapter 4, where a finite number of $M|M|1$ queues is interconnected to form an open queueing network, with feedback. Customers then travel from queueing station to queueing station in order to complete. A transition in such a JQN can either be the arrival of a new customer, the departure of a customer from the JQN or the routing of one customer from one queueing station to another one.

Note that the simplest QBD coincides with the simplest JQN, as both are just a single birth-death process with constant rates, i.e. , an $M|M|1$ queue.

2.2 Paths

While the generator matrix only considers the one-step behavior of the CTMC, the actual evolution of the CTMC over time is specified in detail with a path. In a path states and transitions alternate, where the rates between any two successive states have to be positive to assure that the path can actually be taken. Note that the definition of paths is exactly the same for finite and infinite-state CTMCs.

Definition 7 (Infinite paths). Let $\mathcal{M} = (\mathsf{S}, \mathbf{T}, L)$ be a CTMC. An *infinite path* σ is a sequence $\mathsf{s}_0 \xrightarrow{t_0} \mathsf{s}_1 \xrightarrow{t_1} \mathsf{s}_2 \xrightarrow{t_2} \ldots$ with, for $i \in \mathbb{N}$, $\mathsf{s}_i \in \mathsf{S}$ and $t_i \in \mathbb{R}_{>0}$ such that $\mathbf{T}(\mathsf{s}_i, \mathsf{s}_{i+1}) > 0$ for all i. A *finite path* σ of length $l+1$ is a sequence $\mathsf{s}_0 \xrightarrow{t_0} \mathsf{s}_1 \xrightarrow{t_1} \ldots \mathsf{s}_{l-1} \xrightarrow{t_{l-1}} \mathsf{s}_l$ such that s_l is absorbing, and $\mathbf{T}(\mathsf{s}_i, \mathsf{s}_{i+1}) > 0$ for all $i < l$. $\square$

For an infinite path σ, $\sigma[i] = \mathsf{s}_i$ denotes for $i \in \mathbb{N}$ the $(i+1)$st state of path σ. The time spent in state s_i is denoted by $\delta(\sigma, i) = t_i$. Moreover, with i the smallest index with $t \leq \sum_{j=0}^{i} t_j$, let $\sigma@t = \sigma[i]$ be the state occupied at time t. For finite paths σ with length $l+1$, $\sigma[i]$ and $\delta(\sigma, i)$ are defined in the way described above for $i < l$ only and $\delta(\sigma, l) = \infty$ and $\delta@t = \mathsf{s}_l$ for $t > \sum_{j=0}^{l-1} t_j$. $Path^{\mathcal{Q}}(\mathsf{s})$ is the set of all finite and infinite paths of the CTMC $\mathcal{Q}$ that start in state s and $Path^{\mathcal{Q}}$ includes all (finite and infinite) paths of the CTMC $\mathcal{Q}$.

Now we need a way to state the probability for a given path to be taken while time proceeds. In order to define such a probability measure $\Pr_\alpha$, for an initial distribution α on paths, we need to define *cylinder sets* first.

The cylinder set is a set of paths that is defined by a sequence of states and time intervals of a given length k. The cylinder set then consists of all paths that visit the states stated in the defining sequence in the right order and that change states during the specified time intervals. Thus, the first k states of the paths in the cylinder fit into the special structure specified through the sequence of states and time intervals, while the further behavior remains unspecified.

Definition 8 (Cylinder set). Let $\mathsf{s}_0, \ldots, \mathsf{s}_k \in \mathsf{S}$ be a sequence of states with positive rates $\mathbf{T}(\mathsf{s}_i, \mathsf{s}_{i+1}) > 0$ for $(0 \leq i < k)$, and let $I_0, \ldots, I_{k-1}$ be nonempty intervals in $\mathbb{R}_{\geq 0}$. Then the *cylinder set* $C(\mathsf{s}_0, I_0, \mathsf{s}_1, I_1, \ldots, I_{k-1}, \mathsf{s}_k)$ consists of all paths in $\sigma \in Path^{\mathcal{M}}(\mathsf{s}_0)$ such that $\sigma[i] = \mathsf{s}_i$ for $i \leq k$ and $\delta(\sigma, i) \in I_i$ for $i < k$. ☐

Let α be an initial distribution of the CTMC. Then the probability measure $\Pr_\alpha$ on cylinder sets is defined by induction on the length of the defining sequence k, as follows:

$$\begin{array}{ll} \text{Basis:} & \Pr_\alpha(C(\mathsf{s}_0)) = \alpha(\mathsf{s}_0) \\ \text{Induction step:} & \Pr_\alpha(C(\mathsf{s}_0, I_0, \ldots, \mathsf{s}_k, I', \mathsf{s}')) = \\ & \Pr_\alpha(C(\mathsf{s}_0, I_0, \ldots, \mathsf{s}_k)) \cdot \mathbf{N}(\mathsf{s}_k, \mathsf{s}') \cdot (e^{-\mathbf{E}(\mathsf{s}_k)\cdot a} - e^{\mathbf{E}(\mathsf{s}_k)\cdot b}), \end{array}$$

with $k > 0$, and $a = \inf I'$ and $b = \sup I'$. If s is the only possible initial state ($\alpha(\mathsf{s}) = 1$), we write $\Pr_{\mathsf{s}}$. Recall that $\mathbf{N}(\mathsf{s}_k, \mathsf{s}')$ is the one step probability in the embedded discrete-time Markov chain, as defined in 2.2. For more details on the probability measure on paths refer to [8] and [21].

2.3 Probabilities

Based on the probability measure on paths, two different types of state probabilities can be distinguished for CTMCs. Transient state probabilities are presented in Section 2.3.1 and the steady-state probabilities are presented in Section 2.3.2.

2.3.1 Transient state probability

The *transient state probability* is a time-dependent measure that considers the CTMC $\mathcal{M}$ at a given time instant t. The probability to be in state s' at time instant t, given initial state s, is denoted as:

$$\mathbf{V}^{\mathcal{M}}(\mathsf{s}, \mathsf{s}', t) = \Pr(\sigma \in Path(\mathsf{s}) \mid \sigma@t = \mathsf{s}').$$

The transient probabilities are characterized by a linear system of differential equations of possibly infinite size. Let $\mathbf{V}(t)$ be the matrix of transient state probabilities at time t for all possible starting states s and for all possible goal states s' (we omit the superscript $\mathcal{M}$ for brevity here), then the so-called *Kolmogorov's forward equations* [38]:

$$\frac{d}{dt}\mathbf{V}(t) = \mathbf{V}(t) \cdot \mathbf{Q},$$

describe the transient probabilities, where the initial probabilities are given as $\mathbf{V}(0)$. For finite CTMCs the system of equations can be solved for example with a Taylor series expansion or more efficiently with uniformization [36], also known as Jensen's method [34]. For infinite-state CTMCs, using a standard differential equation solver is impossible since the number of differential equations is infinite. In Chapter 3, we propose a technique called uniformization with representatives, which deals in an efficient way with this differential equation system of infinite size for QBDs. A similar method is developed for JQNs in Chapter 4. We discuss other approaches to compute transient probabilities in Section 5.

2.3.2 Steady-state probability

The *steady-state probabilities* to be in state s', given initial state s, are defined as

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

and indicate the probabilities to be in some state s' "in the long run". Furthermore, if the CTMC is strongly connected, the initial state does not influence the steady-state probabilities (we therefore often write $\pi(\mathsf{s}')$ instead of $\pi(\mathsf{s}, \mathsf{s}')$ for brevity). The steady-state probability vector π then follows from the possibly infinite system of linear equations and its normalization:

$$\pi \cdot \mathbf{Q} = 0, \text{ and } \sum_{\mathsf{s}} \pi_{\mathsf{s}} = 1.$$

For finite CTMCs this system of linear equations can be solved with numerical means known from linear algebra [77]. For infinite-state CTMCs that exhibit a special structure in their state space, this structure can often be exploited to solve the infinite system of linear equations. For QBDs this system of equations can be solved using so-called matrix-geometric methods which exploit the repetitive structure in the matrix $\mathbf{Q}$ as explained in Appendix A. In the context of JQNs the steady-state probabilities can be computed using so-called product-forms as presented in [46, 47]. More details on these methods in general can be found in [61].

2.4 Continuous stochastic logic CSL

Now that we have defined labeled CTMCs we need a formalism to specify desirable properties on states and paths. This can be done with the continuous stochastic logic (CSL) [5], [8], which is a stochastic extension of CTL [20].

In the following we apply the logic CSL [8] on infinite-state CTMCs. The syntax and semantics are the same as for finite CTMCs, with the only difference that we now interpret the formulas over states and paths of infinite-state CTMCs. Therefore, we introduce the syntax and semantics on CTMCs in general.

Definition 9 (CSL). Let $p \in [0,1]$ be a real number, $\bowtie \in \{\leq, <, >, \geq\}$ a comparison operator, $I \subseteq \mathbb{R}_{\geq 0}$ a nonempty interval and AP a set of atomic propositions with $ap \in AP$. *CSL state formulas* Φ are defined by

$$\Phi ::= \mathtt{tt} \mid ap \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{S}_{\bowtie p}(\Phi) \mid \mathcal{P}_{\bowtie p}(\phi),$$

where ϕ is a *CSL path formula* defined on

$$\phi ::= \mathcal{X}^I \Phi \mid \Phi\, \mathcal{U}^I \Phi. \qquad \square$$

For a CSL state formula Φ and a CTMC $\mathcal{M}$, the satisfaction set $Sat(\Phi)$ contains all states of $\mathcal{M}$ that fulfill Φ. Satisfaction is stated in terms of a satisfaction relation, denoted $\models$, as follows.

Definition 10 (Satisfaction on state formulas). The relation $\models$ for states and CSL state formulas is defined as:

$$\begin{array}{llll}
\mathsf{s} \models \mathtt{tt} & \text{for all } \mathsf{s} \in \mathsf{S}, & \mathsf{s} \models \Phi \wedge \Psi & \text{iff } \mathsf{s} \models \Phi \text{ and } \mathsf{s} \models \Psi, \\
\mathsf{s} \models ap & \text{iff } ap \in L(\mathsf{s}), & \mathsf{s} \models \mathcal{S}_{\bowtie p}(\Phi) & \text{iff } \pi^{\mathcal{M}}(\mathsf{s}, Sat(\Phi)) \bowtie p, \\
\mathsf{s} \models \neg\Phi & \text{iff } \mathsf{s} \not\models \Phi, & \mathsf{s} \models \mathcal{P}_{\bowtie p}(\phi) & \text{iff } Prob^{\mathcal{M}}(\mathsf{s}, \phi) \bowtie p,
\end{array}$$

where $\pi^{\mathcal{M}}(\mathsf{s}, Sat(\Phi)) = \sum_{\mathsf{s}' \in Sat(\Phi)} \pi^{\mathcal{M}}(\mathsf{s}, \mathsf{s}')$, and $Prob^{\mathcal{M}}(\mathsf{s}, \phi)$ describes the probability measure of all paths $\sigma \in Path(\mathsf{s})$ that satisfy ϕ when the system is starting in state s, that is, $Prob^{\mathcal{M}}(\mathsf{s}, \phi) = \Pr\{\sigma \in Path^{\mathcal{M}}(\mathsf{s}) \mid \sigma \models \phi\}$. $\square$

The steady-state operator $\mathcal{S}_{\bowtie p}(\Phi)$ denotes that the steady-state probability for Φ-states meets the bound p. $\mathcal{P}_{\bowtie p}(\phi)$ asserts that the probability measure of the paths satisfying ϕ meets the bound p.

Definition 11 (Satisfaction on path formulas). The relation $\models$ for paths and CSL path formulas is defined as:

$$\begin{array}{ll}
\sigma \models \mathcal{X}^I \Phi & \text{iff } \sigma[1] \text{ is defined and } \sigma[1] \models \Phi \text{ and } \delta(\sigma, 0) \in I, \\
\sigma \models \Phi\, \mathcal{U}^I \Psi & \text{iff } \exists t \in I\ (\sigma @ t \models \Psi \wedge (\forall t' \in [0, t)(\sigma @ t' \models \Phi))).
\end{array} \qquad \square$$

We consider the time interval of the next operator to $I = [t_1, t_2]$ for $t_1, t_2 \in \mathbb{R}_{\geq 0}$. The next operator $\mathcal{X}^{[t_1,t_2]}\Phi$ then states that a transition to a Φ-state is made during the time interval $[t_1, t_2]$. The until operator $\Phi\ \mathcal{U}^I \Psi$ asserts that Ψ is satisfied at some time instant $t \in I$ and that at all preceding time instants Φ holds.

In the following, we deal with five different time intervals for the until operator:

- the bounded until operator with interval $I = [0, t]$ for $t \in \mathbb{R}_{>0}$,
- the time interval until with $I = [t_1, t_2]$ for $t_1, t_2 \in \mathbb{R}_{>0}$ and $t_1 < t_2$,
- the point interval until with $I = [t, t]$ for $t \in \mathbb{R}$,
- the unbounded until operator with interval $I = [0, \infty)$ and
- the unbounded until operator with $I = [t, \infty)$ for $t \in \mathbb{R}_{>0}$.

Note that the path formula $\Phi\ \mathcal{U}^I\ \Psi$ is not satisfiable for $I = \varnothing$. For a more detailed description of CSL, see [8].

2.5 Model checking finite state CTMCs

Baier et al. recently proposed numerical methods for model checking CSL formulas over finite state CTMCs [8]. We briefly rehearse the approach developed there, as it forms the basis for our model checking approach for infinite-state CTMCs.

To model check the next operator $\varphi = \mathcal{X}^I\Phi$ we need the one step probabilities to reach a state that fulfills Φ within a time in I.

Proposition 1 (Next operator [8]). For $\mathsf{s} \in \mathsf{S}$, interval $I \subseteq \mathbb{R}_{\geq 0}$ and a CSL state formula Φ:

$$Prob(\mathsf{s}, \mathcal{X}^I\Phi) = (e^{-\mathbf{E}(\mathsf{s})\cdot \inf I} - e^{-\mathbf{E}(\mathsf{s})\cdot \sup I}) \cdot \sum_{\mathsf{s}' \models \Phi} \frac{\mathbf{T}(\mathsf{s}, \mathsf{s}')}{\mathbf{E}(\mathsf{s})}. \qquad \square$$

In [8], it is shown that model checking the time bounded until, the interval until, and the point interval until can be reduced to the problem of computing transient probabilities for CTMCs. The idea is to use a transformed CTMC where several states are made absorbing. As introduced in [8] this proceeds as follows:

Definition 12 (Absorbing). For CTMC $\mathcal{M} = (S, \mathbf{T}, L)$ and CSL state formula Φ let CTMC $\mathcal{M}[\Phi]$ result from $\mathcal{M}$ by making all Φ states in $\mathcal{M}$ absorbing, i.e., $\mathcal{M}[\Phi] = (S, \mathbf{T}', l)$, where $\mathbf{T}'(\mathsf{s}, \mathsf{s}') = \mathbf{T}(\mathsf{s}, \mathsf{s}')$ if $\mathsf{s} \not\models \Phi$ and 0 otherwise. $\square$

The CSL path formula $\varphi = \Phi\ \mathcal{U}^{[0,t]}\Psi$ is valid if a Ψ state is reached, before time t via some Φ path (that is a path via only Φ states). As soon as a Ψ state is reached, the future behavior of the CTMC is irrelevant for the validity of φ. Thus all Ψ states can be made absorbing without affecting the satisfaction set of formula φ. As soon as a $(\neg\Phi \wedge \neg\Psi)$ state is reached, φ will be invalid, regardless of the future evolution of the system. As a result we may switch from $\mathcal{M}$ to $\mathcal{M}[\Psi][\neg\Phi\wedge\neg\Psi] = \mathcal{M}[\neg\Phi\vee\Psi]$, as explained in [8].

Proposition 2 (Time bounded until [8]). For any CTMC $\mathcal{M}$:

$$\mathit{Prob}^{\mathcal{M}}(\mathsf{s}, \Phi\,\mathcal{U}^{[0,t]}\Psi) = \mathit{Prob}^{\mathcal{M}[\Psi]}(\mathsf{s}, \Phi\,\mathcal{U}^{[0,t]}\Psi) = \sum_{\mathsf{s}'\models\Psi} \pi^{\mathcal{M}[\neg\Phi\vee\Psi]}(\mathsf{s}, \mathsf{s}', t). \qquad \square$$

For the interval until with time bound $I = [t_1, t_2], 0 < t_1 \leq t_2$ we again follow the idea of CSL model checking. It is important to note that

$$\mathit{Prob}(\mathsf{s}, \Phi\,\mathcal{U}^{[t_1,t_2]}\Psi) \neq \mathit{Prob}(\mathsf{s}, \Phi\,\mathcal{U}^{[0,t_2]}\Psi) - \mathit{Prob}(\mathsf{s}, \Phi\,\mathcal{U}^{[0,t_1]}\Psi).$$

For model checking a CSL formula that contains the interval Until operator, we need to consider all possible paths, starting in a Φ state at the actual time-instance and reaching a Ψ state during the time interval $[t_1, t_2]$ by only visiting Φ states on the way. We can split such paths in two parts: the first part models the path from the starting state s to a Φ state s' and the second part the path from s' to a Ψ state s'' only via Φ states. We therefore need two transformed CTMCs: $\mathcal{M}[\neg\Phi]$ and $\mathcal{M}[\neg\Phi\vee\Psi]$, where $\mathcal{M}[\neg\Phi]$ is used in the first part of the path and $\mathcal{M}[\neg\Phi\vee\Psi]$ in the second. In the first part of the path, we only proceed along Φ states, thus all states, that do not satisfy Φ do not need to be considered and can be made absorbing. As we want to reach a Ψ state via Φ states in the second part, we can make all state that do not fulfill Φ absorbing, because we cannot proceed along these states, and all states that fulfill Ψ, because we are done, as soon as we reach such a state.

In order to calculate the probability for such a path, we accumulate the multiplied transition probabilities for all triples $(\mathsf{s}, \mathsf{s}', \mathsf{s}'')$, where $\mathsf{s}' \models \Phi$ and is reached before time t_1 and $\mathsf{s}'' \models \Psi$ and is reached before time $t_2 - t_1$. This can be done, because we use CTMCs that are time homogeneous.

Proposition 3 (Interval until [8]). For any CTMC $\mathcal{M}$ and $(0 < t_1 < t_2)$:

$$\mathit{Prob}^{\mathcal{M}}(\mathsf{s}, \Phi\,\mathcal{U}^{[t_1,t_2]}\Psi) = \sum_{\mathsf{s}'\models\Phi}\sum_{\mathsf{s}''\models\Psi} \pi^{\mathcal{M}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1)\cdot\pi^{\mathcal{M}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1). \qquad \square$$

The point interval until can then be seen as a simplification of the interval until, where the second part of the computation does not need to be considered. The CSL path formula $\varphi = \Phi\,\mathcal{U}^{[t,t]}\Psi$ is valid if a Ψ state is reached, at time t via only Φ states, hence all states that do not satisfy Φ do not need to be considered and can be made absorbing. In the goal state s' both Φ and Ψ have to be valid.

Proposition 4 (Point interval until [8]). For any CTMC $\mathcal{M}$:

$$Prob^{\mathcal{M}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,t]}\Psi) = Prob^{\mathcal{M}[\neg\Phi]}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,t]}\Psi) = \sum_{\mathsf{s}'' \models \Phi\wedge\Psi} \pi^{\mathcal{M}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t). \qquad \square$$

In Chapter 3 we show how this concept can be translated to QBDs and in Chapter 4 we present how this approach operates on JQNs for the different intervals.

2.6 General model checking routine

One possibility for model checking that we are going to use is to develop the satisfaction set $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ for a given CSL formula Φ. For every state $s \in S$ it can then be checked whether $s \models \Phi$ by verifying whether $s \in Sat(\Phi)$.

Algorithm 1 $Sat(\Phi$: CSL state formula) : **set of** states

```
begin
if $\Phi = tt$ then
  return $S$;
else if $\Phi \in AP$ then
  return $\{s \in S \mid \Phi \in L(s)\}$;
else if $\Phi = \Phi_1 \wedge \Phi_2$ then
  return $Sat(\Phi_1) \cap Sat(\Phi_2)$;
else if $\Phi = \neg\Phi_1$ then
  return $S \backslash Sat(\Phi_1)$;
else if $\Phi = \mathcal{S}_{\bowtie p}(\Phi_1)$ then
  return $Sat_{\mathcal{S}}(\bowtie p, \Phi_1)$;
else if $\Phi = \mathcal{P}_{\bowtie p}(\mathcal{X}^I \Phi_1)$ then
  return $Sat_{\mathcal{X}}(\bowtie p, I, \Phi_1)$;
else if $\Phi = \mathcal{P}_{\bowtie p}(\Phi_1 \mathcal{U}^I \Phi_2)$ then
  return $Sat_{\mathcal{U}}(\bowtie p, I, \Phi_1, \Phi_2)$;
else
  no valid CSL operator;
end if
end
```

The construction of $Sat(\Phi)$ is done recursively and follows the inductive structure of the CSL syntax. A CSL formula Φ is split into its sub-formulas and for every sub-formula the model checker is invoked recursively, as illustrated in Algorithm 1. All seven CSL operators, as addressed in Section 2.4, are covered and a possibly infinite satisfaction set is returned. The satisfaction set resulting from a steady-state formula is denoted $Sat_{\mathcal{S}}$, the satisfaction set resulting from a next formula is denoted $Sat_{\mathcal{X}}$ and the satisfaction set resulting from an until formula $Sat_{\mathcal{U}}$, respectively. The algorithms to compute these satisfaction sets will be introduced in Chapter 3 for QBDs and for JQNs in Chapter 4.

In the following, this set of states will be a special data structure, in order to deal with possibly infinite satisfaction sets. However, the data structure depends on the type of infinite Markov chain that is model checked. We will introduce this data structure for QBDs in Chapter 3 and for JQNs in Chapter 4.

2.7 Summary

In this chapter we introduced the foundations of stochastic model checking. We presented labeled CTMCs with finite and infinite-state space in general, and an infinite CTMC with highly structured state space in particular, namely a birth-death process. We discussed paths on CTMCs and the probability measure on paths that follows from the cylinder set. Furthermore, we introduced two different probability measures on CTMCs, transient and steady-state probabilities. The syntax and semantics of CSL have been shown to be the same on finite and on infinite-state CTMCs. We discussed a general model checking routine based on satisfaction sets and gave an overview on how the next and the until operator are model checked on finite CTMCs.

Chapter 3

CSL model checking algorithms for QBDs

In this chapter we describe CSL model checking algorithms for labeled QBDs. First, the class of labeled QBD processes is introduced in Section 3.1. General algorithms for CSL model checking of QBDs are then presented in Section 3.2. Section 3.3 presents uniformization for QBDs as needed for transient analysis of QBDs [70]. The details of model checking the until operator with its different time bounds are described in Section 3.4. A small case study is presented in Section 3.5, before we conclude in Section 3.6.

3.1 Labeled Quasi Birth Death processes

A special case of infinite-state CTMCs are CTMCs with so-called quasi birth-death structure [61]. The infinite state space of a QBD can be viewed as a two-dimensional strip, which is finite in one dimension and infinite in the other. The states in this strip are grouped in so-called *levels*, according to their identity in the infinite dimension. Figure 3.1 gives a graphical representation of a QBD.

Definition 13 (Labeled QBD). A **labeled QBD** $\mathcal{Q}$ of order (N_0, N) (with $N_0, N \in \mathbb{N}^+$) is a labeled infinite-state continuous-time Markov chain, defined as a tuple $(\mathsf{S}, \mathbf{T}, L)$ with an infinite countable set of states $\mathsf{S} \subset \mathbb{N}^2$, a transition rate matrix $\mathbf{T} : \mathsf{S} \times \mathsf{S}$ and the labeling function $L : \mathsf{S} \to 2^{AP}$.

Transitions, represented by positive entries in $\mathbf{T}$, can only occur between states of the same level or between states of neighboring levels. Level 0 is called *boundary level*, level 1 is called *border level* and all levels at least 1 are called repeating levels. All repeating levels have the same inter-level and intra-level transition structure.

The block-tridiagonal generator matrix $\mathbf{Q} : \mathsf{S} \times \mathsf{S} \to \mathbb{R}_{\geq 0}$ that is computed by removing possible self loops from $\mathbf{T}$, consists of the following finite matrices describing the inter- and intra-level transitions, as shown in Figure 3.2:

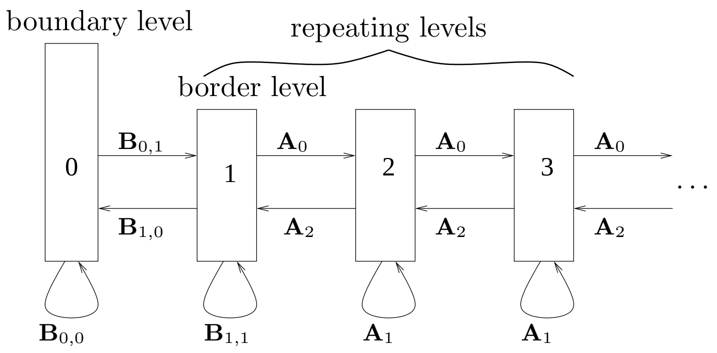


Figure 3.1: Sketch of the state space of a QBD

- $\mathbf{B}_{0,0} \in \mathbb{R}^{N_0 \times N_0}$: intra-level transition structure of the boundary level,
- $\mathbf{B}_{0,1} \in \mathbb{R}^{N_0 \times N}$: inter-level transitions from the boundary level to the border level,
- $\mathbf{B}_{1,0} \in \mathbb{R}^{N \times N_0}$: inter-level transitions from the border level to the boundary level,
- $\mathbf{B}_{1,1} \in \mathbb{R}^{N \times N}$: intra-level transition structure of the border level,
- $\mathbf{A}_0 \in \mathbb{R}^{N \times N}$: inter-level transitions from one repeating level to the next higher repeating level,
- $\mathbf{A}_1 \in \mathbb{R}^{N \times N}$: intra-level transitions for the repeating levels, and
- $\mathbf{A}_2 \in \mathbb{R}^{N \times N}$: inter-level transitions from one repeating level to the next lower repeating level.

□

Note that $\mathbf{B}_{1,1}$ differs from $\mathbf{A}_1$ only in the diagonal entries. From a fixed set AP of atomic propositions the labeling function $L : \mathsf{S} \to 2^{AP}$ assigns to each state the set of valid atomic propositions in that state.

The set of states S can be partitioned into an infinite number of finite sets $S^j, j = \{0, 1, \cdots\}$, each containing the states of one level, such that $\mathsf{S} = \bigcup_{j=0}^{\infty} S^j = \{0, \cdots, N_0 - 1\} \times \{0\} \cup \{0, \cdots, N-1\} \times \mathbb{N}^+$, where the first part represents the boundary level with N_0 states, and the second part the infinite number of repeating levels, each with N states. We call the first repeating level the border level. Two states (i_1, j_1) and (i_2, j_2) are called *corresponding states* if $i_1 = i_2$, $j_1, j_2 > 0$ and $j_1 \neq j_2$.

The states of each level S^i for $i > 0$ are divided into three, not necessarily disjoint, sets of states: $S^i = S_{in}^{i,\uparrow} \cup S_{center}^{i,\uparrow} \cup S_{out}^{i,\uparrow}$.

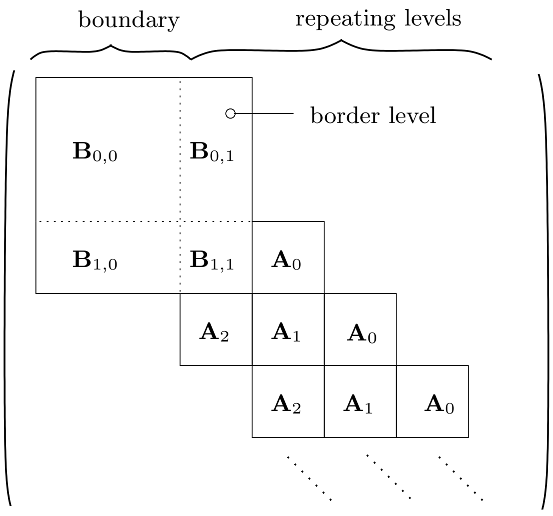


Figure 3.2: Generator matrix for a QBDs

- The set $S^{i,\uparrow}_{in}$ comprises states that can be reached from the next lower level $(i-1)$ in one step, $S^{i,\uparrow}_{center}$ comprises the states from which level $i+1$ cannot be reached in one step, and $S^{i,\uparrow}_{out}$ comprises the states from which the next higher level $(i+1)$ can be reached in one step.

- Similarly, we define $S^{i,\downarrow}_{in}$ to comprise the states that can be reached from the next higher level in one step, $S^{i,\downarrow}_{center}$ to comprise the states from which level $i-1$ cannot be reached in one step and $S^{i,\downarrow}_{out}$ to comprise all states from which the next lower level can be reached in one step.

Note that for the boundary level we have $S^0 = S^{0,\uparrow}_{center} \cup S^{0,\uparrow}_{out}$ and $S^0 = S^{0,\downarrow}_{center} \cup S^{0,\downarrow}_{in}$, because $S^{0,\uparrow}_{in} = \varnothing$ and $S^{0,\downarrow}_{out} = \varnothing$. The minimum number of steps that has to be undertaken to reach s_2 from s_1 is given by $g(\mathsf{s}_1, \mathsf{s}_2) = |\mathit{shortestpath}(\mathsf{s}_1, \mathsf{s}_2)|$. Let $d^{\uparrow} \geq 1$ be the so-called *upward level diameter*, that is, the minimum number of state transitions needed to reach the next higher repeating level from a state in $S^{i,\uparrow}_{in}$: $d^{\uparrow} = \min\{g(\mathsf{s}_1, \mathsf{s}_2) \mid \mathsf{s}_1 \in S^{i,\uparrow}_{in}, \mathsf{s}_2 \in S^{i+1,\uparrow}_{in}\}$. The *downward level diameter* $d^{\downarrow}$ is defined along the same lines as $d^{\downarrow} = \min\{g(\mathsf{s}_1, \mathsf{s}_2) \mid \mathsf{s}_1 \in S^{i,\downarrow}_{in}, \mathsf{s}_2 \in S^{i-1,\downarrow}_{in}\}$. We define d, the *symmetric level diameter*, as the minimum of the upward and downward level diameter. Because the repeating levels of a QBD all exhibit the same structure, they all have the same level diameter. However, the number of steps needed to cross l levels may be larger than $l \cdot d$, depending on the structure of the QBD.

Example 1. To illustrate the concept of the level diameter, Figure 3.3 shows three successive levels of a QBD with five states per level. We derive the upwards and the downwards level diameter by arranging the states of one level into the different sets, as explained above.

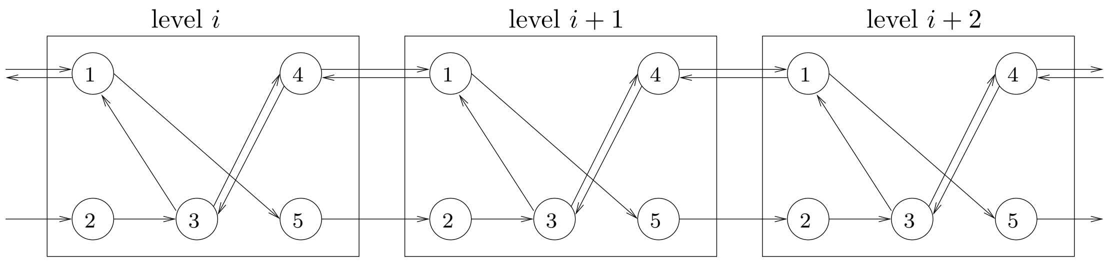


Figure 3.3: Three successive levels of a QBD to illustrate the concept of level diameter

The set $S_{in}^{i,\uparrow}$ comprises the states $\{(1,i),(2,i)\}$, the set $S_{center}^{i,\uparrow}$ comprises the states $\{(1,i),(2,i),(3,i)\}$ and the set $S_{out}^{i,\uparrow}$ comprises the states $\{(4,i),(5,i)\}$. Starting in state $(1,i) \in S_{in}^{i,\uparrow}$ and ending in state $(2,i+1) \in S_{in}^{i+1,\uparrow}$, yields the minimum number of transitions to reach the next higher level. Hence, the upwards level diameter $d^{\uparrow}$ is set to 2.

For the downwards level diameter, $S_{in}^{i,\downarrow} = \{(4,i)\}$, the set $S_{center}^{i,\downarrow}$ comprises the states $\{(2,i),(3,i),(4,i),(5,i)\}$ and the set $S_{out}^{i,\downarrow} = \{(1,i)\}$. Starting in state $(4,i+1) \in S_{in}^{i+1,\downarrow}$ and ending in state $(4,i) \in S_{in}^{i,\downarrow}$, yields the minimum number of transitions to reach the next lower level. Hence, the downwards level diameter $d^{\downarrow}$ equals 3. Consequently, the symmetric level diameter in this QBD is set to 2.

Clearly, for crossing the next two lower levels, more than $d \cdot 2$ steps are needed, as the downwards level diameter is higher than the symmetric level diameter. However, due to the special structure of the QBD, we also need more than $d \cdot 2$ steps to cross the next two higher levels. Since we always have to cross one of the two levels via the longer path that contains state $(3,i)$, we need 5 steps to cross the next two higher levels.

3.2 Model checking algorithms

In this section we present the general algorithms for CSL model checking QBDs. Section 3.2.1 introduces the concept of level independence for atomic properties. In Section 3.2.2 this is extended to CSL formulas in general. We present the general model checking routine for QBDs in Section 3.2.3. How to model check atomic properties and logical operators is presented in Section 3.2.4. Model checking the steady-state operator is introduced in Section 3.2.5, model checking the next operator in Section 3.2.6 and model checking the different until operators in Section 3.2.7.

3.2.1 Level independent atomic properties

In the following we limit ourselves to strongly connected QBDs with so-called *level independent* atomic propositions. That is, if an atomic proposition $ap \in AP$ is valid in a certain state of an arbitrary repeating level, it has to be valid in the corresponding states of all repeating levels. This limitation poses a true restriction on the set of formulas we are able to check. In practice, this means that atomic propositions must not refer to the level index in order to be level independent.

Definition 14 (Level independent atomic proposition). Let $i \in \{0, \dots, N-1\}$, an atomic proposition $ap \in AP$ is *level independent* if

$$\text{for all } l, k \geq 1, ap \in L(i,k) \Leftrightarrow ap \in L(i,l). \qquad \square$$

In order to develop efficient CSL model checking algorithms for QBDs, we need to exploit the connection between the validity of state formulas and the special structure of QBDs. At first glance one could think that in corresponding states of all repeating levels the same CSL formulas hold. Unfortunately this is not the case, which can easily be seen when considering the time-bounded next operator. In the border level different next-formulas might be satisfied than in the other repeating levels, because the boundary level is still reachable from the border level but not from any other repeating level. Thus, if we want to check for example the formula $\phi = \mathcal{X}^{[t_1,t_2]} red$ and the property *red* is only valid in the boundary level, this property ϕ can be fulfilled by a path starting in the border level, but not when starting in any other repeating level. A similar reasoning holds for the until operator, where not only the border level is concerned but even more repeating levels, because with the until operator not just one step is considered, but potentially an infinite number. Thus, no two repeating levels can *a priori* be considered to satisfy the same path-formulas.

3.2.2 Level independence of CSL formulas

Even though CSL formulas are not level independent in general, their validity does not change arbitrarily between levels. Remember that we assume level independence of atomic propositions for the QBDs we consider. For CSL formulas, we generalize the idea of level independence: we show that the validity in a state is level independent for repeating levels with an index of at least k for some $k > 0$. Thus, the validity of a CSL formula changes between corresponding states of repeating levels, but only up to repeating level $k - 1$. From level k onwards, the validity remains unchanged.

Definition 15 (Level independence of CSL formulas). Let $\mathcal{Q}$ be a QBD of order (N_0, N). A CSL state formula Φ is *level independent as of level* $k \geq 1$ (in QBD $\mathcal{Q}$) if and only if for levels above and including k, the validity of Φ in a state does not depend on the level, that is, for all $i \in \{0, \ldots, N-1\}$ and for all $l \geq k$: $(i, l) \models \Phi \iff (i, k) \models \Phi$. $\square$

The following proposition states, under the assumption of level independent atomic propositions, that such a k exists for any CSL state formula. We will justify this proposition inductively over the structure of the logic: in Section 3.2.4 for atomic propositions and logical operators, in Section 3.2.5 for the steady-state operator, in Section 3.2.6 for the next operator and in Section 3.4.6 for the different until operators.

Note that atomic propositions do not have to be level-independent as of level 1. In case the atomic propositions are level independent as of level k, we just extend the boundary level to the first $k-1$ repeating levels.

Proposition 5 (Level independence on QBDs). Let $\mathcal{Q}$ be a QBD with level independent atomic propositions and let Φ be a CSL state formula other than $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^I \Psi)$. Then there exists a $k \in \mathbb{N}$, such that Φ is level independent as of level k in $\mathcal{Q}$.

For the until operator $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^I \Psi)$ we require that for no state s the probability measure is exactly equal to p, hence, $Prob(\mathsf{s}, \Phi \mathcal{U}^I \Psi) \neq p$. Under this assumption, there exists a $k \in \mathbb{N}$, such that $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^I \Psi)$ is level independent as of level k in $\mathcal{Q}$. $\square$

Furthermore, we assume that the QBDs under study are strongly connected, because it simplifies checking the steady-state operator, as we do not have to consider several parts of the QBD.

3.2.3 General model checking

For model checking a property Φ, we compute the set $Sat(\Phi)$ with the recursive descent procedure over the parse tree of Φ, as presented in Section 2.6. For a state formula Φ that is level independent as of level k, only the first k level satisfaction sets have to be computed.

Definition 16 (Level i satisfaction set Sat^i). Given a CSL state formula Φ, we define the satisfaction set of level i as $Sat^i(\Phi)$ and the possibly infinite satisfaction set $Sat(\Phi)$ can then be expressed as the union over all level satisfaction sets:

$$Sat^i(\Phi) = Sat(\Phi) \cap S^i \text{ and } Sat(\Phi) = \bigcup_{i=0}^{\infty} Sat^i(\Phi). \qquad \square$$

Given Φ is level independent as of level k, $Sat^k(\Phi)$ acts as a representative for all levels above k as the validity of Φ does not change any more for higher levels. Thus, for a CSL formula Φ that is level independent as of k, we do not need to consider the possibly infinite satisfaction set $Sat(\Phi)$; it suffices to consider the level satisfaction sets up to level k: $\bigcup_{i=0}^{k} Sat^{i}(\Phi)$. For QBDs, the satisfaction sets, as used in Chapter 2, Algorithm 1, can therefore be represented by the data structure $\bigcup_{i=0}^{k} Sat^{i}(\Phi)$ that contains all states that fulfill Φ up to level k, in combination with the information that Φ is level independent as of k.

3.2.4 Atomic propositions and logical operators

Computing the satisfaction set for an atomic proposition ap proceeds as follows. $Sat^{0}(ap)$ consists of those states of the boundary level where ap is contained in the labeling. We model check all states in the border level in order to obtain $Sat^{1}(ap)$, and, similarly, $Sat^{j}(ap)$ for $j \geq 1$.

Let Φ be a CSL state formula that is level independent as of level k. Its negation $\neg\Phi$ is clearly also level independent as of level k. The level satisfaction sets of $\neg\Phi$ are computed by complementing the corresponding satisfaction set of Φ: $Sat^{j}(\neg\Phi) = S^{j} \backslash Sat^{j}(\Phi), \text{for all } j \geq 0$.

Let Φ and Ψ be two CSL state formulas, level independent as of level k_Φ and k_Ψ, respectively. The conjunction $\Phi \wedge \Psi$ is level independent as of level $\max(k_\Phi, k_\Psi)$. The level satisfaction sets are computed by intersecting the corresponding satisfaction sets of Φ and Ψ: $Sat^{j}(\Phi \wedge \Psi) = Sat^{j}(\Phi) \cap Sat^{j}(\Psi), \text{for all } j \geq 0$. The level satisfaction set $Sat^{\max(k_\Phi, k_\Psi)}(\Phi \wedge \Psi)$ is the representative for all following levels.

3.2.5 Steady-state operator

A state s satisfies $\mathcal{S}_{\bowtie p}(\Phi)$ if the sum of the steady-state probabilities of all Φ-states reachable from s meets the bound p. Since we assume a strongly connected QBD, the steady-state probabilities are independent of the starting state. It follows that either all states satisfy a steady-state formula or none of the states does, which implies that a steady-state formula is always level independent as of level 1, since the boundary level may have a different structure. We first determine the satisfaction set $Sat(\Phi)$ and then compute the accumulated steady-state probability. If the accumulated steady-state probability meets the bound p, we have $Sat(\mathcal{S}_{\bowtie p}(\Phi)) = \mathsf{S}$, otherwise, $Sat(\mathcal{S}_{\bowtie p}(\Phi)) = \varnothing$.

Exploiting the special structure of QBDs, the accumulated probability is given by

$$\pi(Sat(\Phi)) = \sum_{\mathsf{s} \in Sat(\Phi)} \pi(\mathsf{s}) = \sum_{j=0}^{\infty} \sum_{\mathsf{s} \in Sat^{j}(\Phi)} \pi_j(\mathsf{s}),$$

where the vectors $\pi_j = (\cdots, \pi_j(\mathsf{s}), \cdots)$ can be computed one after the other, using the matrix-geometric method, cf. [61], as explained in Appendix A. In order to deal with the infinite summation we iterate through the repeating levels and accumulate the steady-state probabilities in a level-wise fashion. We denote with $\tilde{\pi}^l(Sat(\Phi))$ the accumulated steady-state probabilities of all Φ-states up to level l, that is,

$$\tilde{\pi}^l(Sat(\Phi)) = \sum_{j=0}^{l} \sum_{\mathsf{s} \in Sat^j(\Phi)} \pi_j(\mathsf{s}).$$

Starting with $l = 0$, we compute $\tilde{\pi}^l(Sat(\Phi))$ and $\tilde{\pi}^l(Sat(\neg\Phi))$, respectively. The computation of the steady-state probabilities of $\neg\Phi$-states introduces no additional cost, since we have to compute the whole vector π_j anyway. In every step we have to check whether we can already decide on the validity of the steady-state formula $\mathcal{S}_{\bowtie p}(\Phi)$. The following implications hold:

$$\begin{array}{lrcl} \text{(a)} & \tilde{\pi}^l(Sat(\Phi)) > p & \Rightarrow & \pi(Sat(\Phi)) > p, \\ \text{(b)} & \tilde{\pi}^l(Sat(\neg\Phi)) > 1 - p & \Rightarrow & \pi(Sat(\Phi)) < p. \end{array}$$

As soon as one of the left-hand side inequalities becomes true, we can stop. The model checking routine for the steady-state operator is stated in pseudocode in Algorithm 2.

For the interpretation we distinguish the cases $\mathcal{S}_{<p}(\Phi)$ and $\mathcal{S}_{>p}(\Phi)$. For $\mathcal{S}_{<p}(\Phi)$ the interpretation is as follows. If inequality (a) holds, the condition $\pi(Sat(\Phi)) < p$ is clearly not accomplished and $Sat(\mathcal{S}_{<p}(\Phi)) = \varnothing$. If inequality (b) holds, the condition $\pi(Sat(\Phi)) < p$ is accomplished and $Sat(\mathcal{S}_{<p}(\Phi)) = \mathsf{S}$. As every steady-state formula is independent as of level 1, Algorithm 2 just returns $Sat^0 \cup Sat^1$. How to interpret the termination criterion is presented in pseudocode in Algorithm 3. In case the steady-state formula is valid the algorithm returns $S^0 \cup S^1$ and otherwise it returns two empty sets.

Algorithm 2 $Sat_{\mathcal{S}}(\bowtie p, \Phi) : \bigcup_{i=0}^{1} Sat^i$

```
begin
i = 0;
$sat = S \cap Sat(\Phi)$;
$sat\_neg = S \backslash Sat$;
$\widetilde{\pi}(\Phi) = 0$;
$\widetilde{\pi}(\neg\Phi) = 0$;
while $(\widetilde{\pi}(\Phi) \leq p)$ and $(\widetilde{\pi}(\neg\Phi) \leq 1 - p)$ do
  $\pi_i$ = MGM(level $i$);                    (* according to App. A *)
  $\widetilde{\pi}(\Phi) \mathrel{+}= \sum_{s' \in sat} \pi_i(s')$;
  $\widetilde{\pi}(\neg\Phi) \mathrel{+}= \sum_{s' \in sat\_neg} \pi_i(s')$;
  $i = i + 1$;
end while
return $interpret(\bowtie p, \Phi, \widetilde{\pi}(\Phi), \widetilde{\pi}(\neg\Phi))$;
end
```

Algorithm 3 *interpret*$(\bowtie p, \Phi, \widetilde{\pi}(\Phi), \widetilde{\pi}(\neg\Phi)) : \bigcup_{i=0}^{1} Sat^i$

```
begin
if $\bowtie p = (< p) \vee (\leq p)$ then
  if $\widetilde{\pi}(\Phi) > p$ then
    return $\varnothing$;
  else
    return $S^0 \cup S^1$;
  end if
else
  if $\widetilde{\pi}(\Phi) > p$ then
    return $S^0 \cup S^1$;
  else
    return $\varnothing$;
  end if
end if
end
```

For $\mathcal{S}_{>p}(\Phi)$ the same conditions need to be checked in every iteration step l, but they need to be interpreted differently; if inequality (a) holds, the probability bound is met and $Sat(\mathcal{S}_{>p}(\Phi)) = \mathsf{S}$. If inequality (b) holds, the bound is not met and $Sat(\mathcal{S}_{>p}(\Phi)) = \varnothing$. For $\mathcal{S}_{\geq p}(\Phi)$ or $\mathcal{S}_{\leq p}(\Phi)$ the equations need to be modified accordingly.

The satisfaction set of Φ might be finite. For a CSL formula Φ that is level independent as of level k, this is the case when no state in level k satisfies Φ. The iteration then ends at level $k-1$ and $\pi(Sat(\Phi)) = \tilde{\pi}^{k-1}(Sat(\Phi))$. In case $Sat(\Phi)$ is of infinite size, the iteration stops as soon as one of the inequalities is satisfied. Unfortunately, if the bound p is exactly equal to the steady-state probability $\pi(Sat(\Phi))$, the approximations $\tilde{\pi}^l(Sat(\Phi))$ and $\tilde{\pi}^l(Sat(\neg\Phi))$ will never fulfill one of the inequalities. In an implementation of this algorithm some care must be taken to detect this case in order to avoid a non-termination iteration, for example a maximum iteration bound can be introduced.

Instead of the just-sketched iterative process, we can also use a closed-form matrix expression for the probability $\pi(Sat(\Phi))$ by exploiting properties of the matrix-geometric solution, i.e., by using the fact that $\sum_{i=0}^{\infty} \mathbf{R}^i = (\mathbf{I} - \mathbf{R})^{-1}$, according to [63, Section 4.2]. In doing so, the infinite summation disappears and hence, the termination problem is avoided. Note that the matrix inversion is computed anyway when using the matrix-geometric method, hence this approach is therefore not necessarily less efficient.

3.2.6 Time-bounded next operator

Recall that a state s satisfies $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ if the one-step probability to reach a state that fulfills Φ within a time $t \in [t_1, t_2]$, outgoing from s meets the bound p. As for one-step probabilities self loops have to be taken into account, we have to use the transition rate matrix $\mathbf{T}$ to model check the time-bounded next operator:

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi) \;&\Leftrightarrow\; \Pr\{\sigma \in Path(\mathsf{s}) \mid \sigma \models \mathcal{X}^{[t_1,t_2]}\Phi\} \;\bowtie p \\ &\Leftrightarrow \left(\left(e^{-\mathbf{E}(\mathsf{s})\cdot t_1} - e^{-\mathbf{E}(\mathsf{s})\cdot t_2}\right) \cdot \sum_{\mathsf{s}' \in Sat(\Phi)} \frac{\mathbf{T}(\mathsf{s},\mathsf{s}')}{\mathbf{E}(\mathsf{s})} \right) \bowtie p, \end{aligned} \tag{3.1}$$

where $e^{-\mathbf{E}(\mathsf{s})\cdot t_1} - e^{-\mathbf{E}(\mathsf{s})\cdot t_2}$ is the probability of leaving s at a time $t \in [t_1, t_2]$, and $\mathbf{T}(\mathsf{s},\mathsf{s}')/\mathbf{E}(\mathsf{s})$ specifies the probability to step from state s to state s'. Note that the above inequality contains a summation over all Φ-states. We, however, only need to sum over the states of $Sat(\Phi)$ that are reachable from s in one step. That is, for $\mathsf{s} = (i, j)$, we only have to consider the Φ-states from levels $j-1, j$, and $j+1$; the one-step probabilities for all other states are zero, thus making this summation finite.

Now, let the inner formula Φ of the next-formula be level independent as of level k. Hence, the validity of the state formula $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ might be different in corresponding states for all levels up to $k-1$. Therefore, unfortunately, level k can still have different states satisfying $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ since level $k-1$ is reachable in one step. But, as of level $k+1$, only levels can be reached where the validity of state formula Φ is equal for corresponding states. Hence, if Φ is level independent as of level k, $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ is level independent as of level $k+1$. For the construction of the satisfaction set of such a formula, we therefore have to compute explicitly the satisfying states up to level $k+1$. Subsequently, $Sat^{k+1}(\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi))$ can be seen as a representative for all following repeating levels. That is,

$$Sat^{k+1}(\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)) = Sat^{k+i}(\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)), \text{ for } i > 1,$$

because the validity of $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ does not change anymore from level $k+1$ onwards. Model checking the next operator is stated in pseudocode in Algorithms 4 and 5.

3.2.7 Time-bounded until operator

To model check $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I \Psi)$ for a given state s we adopt the general approach for finite CTMCs [8]. The idea is to use a transformed QBD where several states are made absorbing. Recall, that the CSL path formula $\varphi = \Phi\, \mathcal{U}^I \Psi$ is valid if a Ψ-state is reached on a path during the time interval I via only Φ-states. We discuss model checking the until operator for the intervals $[0, t]$, $[t_1, t_2]$, $[t, t]$, $[0, \infty)$ and $[t, \infty)$

Algorithm 4 $Sat_{\mathcal{X}}(\bowtie p, I, \Phi) : \bigcup_{i=0}^{k+1} Sat^i$

```
begin
$Sat(\Phi)$ independent as of $k$;
for all $i \in \{0, \ldots, k+1\}$ do
  for all $s \in S^i$ do
    if satisfy$_{\mathcal{X}}(s, \bowtie p, I, \Phi)$ then
      $Sat^i = Sat^i \cup \{s\}$;
    end if
  end for
end for
return $\bigcup_{i=0}^{k+1} Sat^i$;
end
```

Algorithm 5 satisfy$_{\mathcal{X}}(s, \bowtie p, I, \Phi)$: **boolean**

```
begin
a = sup(I);
b = inf(I);
return $[(e^{-E(s)\cdot b} - e^{-E(s)\cdot a}) \sum_{s' \in Sat(\Phi)} \frac{\mathbf{T(s,s')}}{\mathbf{-E(s)}}] \bowtie p$;
end
```

and present the connection between these five cases and the involved numerical algorithms to be discussed in Section 3.3. The justification of Proposition 5 for the until operators is postponed to Section 3.4.6, as we need a better understanding of how the probabilities are actually computed first.

Case $I = [0, t]$

First, we restrict the time interval to a time interval $I = [0, t]$. In this case, the future behavior of the QBD is irrelevant for the validity of φ, as soon as a Ψ-state is reached. Thus all Ψ-states can be made absorbing without affecting the satisfaction set of formula φ. On the other hand, as soon as a $(\neg\Phi \wedge \neg\Psi)$-state is reached, φ will be invalid, regardless of the future evolution. As a result we may switch from checking the Markov chain $\mathcal{Q}$ to checking the Markov chain $\mathcal{Q}[\Psi][\neg\Phi \wedge \neg\Psi] = \mathcal{Q}[\neg\Phi \vee \Psi]$, as defined in Chapter 2, where all states satisfying the formula in $[\cdot]$ are made absorbing.

Proposition 6 (Connectivity of absorbing $\mathcal{Q}$). Given a strongly connected QBD $\mathcal{Q}$ and a level-independent CSL formula Φ, the Markov chain $\mathcal{Q}[\Phi]$ is still a QBD, however, $\mathcal{Q}[\Phi]$ is not necessarily strongly connected, anymore. $\square$

Model checking a formula involving the until operator then reduces to calculating the transient probabilities $\pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}, \mathsf{s}', t)$ for all Ψ-states s'. Exploiting the regular structure of QBDs yields

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]} \Psi) &\Leftrightarrow Prob^{\mathcal{Q}}(\mathsf{s}, \Phi\, \mathcal{U}^{[0,t]} \Psi) \;\bowtie\; p \\ &\Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \pi^{\mathcal{Q}[\neg\Phi \vee \Psi]}(\mathsf{s}, \mathsf{s}', t) \right) \bowtie p. \end{aligned} \tag{3.2}$$

The transient probability of being in each state of the infinite-state QBD for any possible initial state (as needed for the time-bounded until operators) can be calculated with a new iterative uniformization-based method, which we present in Section 3.3. To calculate the satisfaction set for $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]} \Psi)$, we need to understand how this algorithm works, therefore we postpone this discussion to Section 3.4.1.

Case $I = [t_1, t_2]$

Considering a time interval $[t_1, t_2]$ with $0 < t_1 < t_2$ we can split the computation in two parts. The first part then addresses the path from the starting state s to a Φ-state s' at time t_1 via only Φ states. The second part of the computation addresses the path from s' to a Ψ-state s'' via only Φ states. This leads us to two transformed QBDs: $\mathcal{Q}[\neg\Phi]$ that is used in the first part (i.e., for the interval $[0, t_1)$) and $\mathcal{Q}[\neg\Phi \vee \Psi]$ in the second part (i.e. for the interval $[t_1, t_2]$). To calculate the probability for such a path, we accumulate the product of the transient probabilities for all triples $(\mathsf{s}, \mathsf{s}', \mathsf{s}'')$, where $\mathsf{s}' \models \Phi$ is reached before time t_1 and $\mathsf{s}'' \models \Psi$ is reached before time $t_2 - t_1$. This can be done, because the QBDs are time homogeneous. Hence, we have:

$$\begin{aligned} &\mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[t_1,t_2]} \Psi) \Leftrightarrow Prob^{\mathcal{Q}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t_1,t_2]} \Psi) \;\bowtie\; p \\ \Leftrightarrow &\left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^j(\Psi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1) \cdot \pi^{\mathcal{Q}[\neg\Phi \vee \Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1) \right) \bowtie p. \end{aligned} \tag{3.3}$$

The algorithm for the interval until will be presented in Section 3.4.3.

Case $I = [t, t]$

The point interval until is a simplification of the interval until, where only the first part of the computation needs to be taken into account. Thus, we need the transformed QBD $\mathcal{Q}[\neg\Phi]$ and need to compute the probability that at time point t a state s' is reached that fulfills $\Phi \wedge \Psi$.

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[t,t]} \Psi) &\Leftrightarrow Prob^{\mathcal{Q}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,t]} \Psi) \;\bowtie\; p \\ &\Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Phi \wedge \Psi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \right) \bowtie p. \end{aligned} \tag{3.4}$$

The algorithm for the point interval until operator is the same as for the time bounded until, with two minor changes. First the transient probabilities have to be computed on $\mathcal{Q}[\neg\Phi]$ for the point interval until and on $\mathcal{Q}[\neg\Phi \vee \Psi]$ for the time bounded until. Second, the goal states s' have to fulfill $\Phi \wedge \Psi$ for the point interval until and just Ψ for the time bounded until.

Case $I = [0, \infty)$

For the unbounded case (interval $[0, \infty)$) the probability $Prob^{Q}(\mathsf{s}, \Phi\, \mathcal{U}^{[0,\infty)}\Psi)$ equals the probability to eventually reach a Ψ-state via only Φ-states. Since the $\neg\Phi \vee \Psi$-states are absorbing, this is exactly the steady-state probability to be in a Ψ-state in the adapted QBD. However, due to the fact that $\mathcal{Q}[\neg\Phi \vee \Psi]$ is not necessarily strongly connected, cf. Proposition 6, we cannot compute the satisfaction set of $\mathcal{P}_{\bowtie p}(\Phi\mathcal{U}^{[0,\infty)}\Psi)$ with the algorithm presented in Section 3.2.5.

$$\begin{aligned} s \models \mathcal{P}_{\bowtie p}(\Phi\mathcal{U}^{[0,\infty)}\Psi) &\Leftrightarrow Prob^{Q}(s, \Phi\, \mathcal{U}^{[0,\infty)}\Psi) \bowtie p \\ &\Leftrightarrow \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}, Sat(\Psi)) \bowtie p \\ &\Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat(\Psi)} \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}, \mathsf{s}') \right) \bowtie p. \end{aligned} \tag{3.5}$$

The algorithm for the unbounded until operator with interval $I = [0, \infty)$ will be discussed in Section 3.4.4.

Case $I = [t, \infty)$

For the interval $[t, \infty)$ the computation is split in two parts, just as for $[t_1, t_2]$. The first part addresses the path from the starting state s to a Φ-state s' via only Φ-states at time t, whereas the second part addresses the path that eventually leads from s' to a Ψ-state. Note that we combine the transient probabilities in the transformed QBD $\mathcal{Q}[\neg\Phi]$ for the first part, with the steady-state probabilities in $\mathcal{Q}[\neg\Phi \vee \Psi]$ for the second part as follows:

$$\begin{aligned} &\mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[t,\infty)}\Psi) \Leftrightarrow Prob^{\mathcal{Q}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,\infty)}\Psi) \ \bowtie \ p \\ \Leftrightarrow &\left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{i}(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{j}(\Psi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'') \right) \bowtie p. \end{aligned} \tag{3.6}$$

The algorithm for the unbounded until operator with interval $I = [t, \infty)$ will be described in Section 3.4.5.

3.3 Uniformization with Representatives

We first describe the main principles of uniformization for QBDs in Section 3.3.1. In Section 3.3.2 we then describe how to exploit the QBD structure to obtain a finite data representation. We address the growth of the involved data structures in Section 3.3.3. The actual iterative algorithm is then presented in Section 3.3.4 before we discuss complexity issues in Section 3.3.5.

3.3.1 Uniformization

Uniformization is a well-established technique to determine the transient-state probabilities $\mathbf{V}(t)$ in a continuous-time Markov chain via an uniformized discrete-time Markov chain subordinated to a Poisson process [36]. The parameter of this Poisson process corresponds to the maximum outgoing transition rate of any single state in the CTMC. This so-called uniformization rate λ can easily be determined because $\mathbf{Q}$ has only a finite number of different diagonal entries (originating from the matrices $\mathbf{B}_{0,0}, \mathbf{B}_{1,1}$, and $\mathbf{A}_1$).

The probability matrix $\mathbf{P}$ for the uniformized DTMC then is computed as $\mathbf{I}+\mathbf{Q}/\lambda$ and it follows the same tridiagonal structure as $\mathbf{Q}$ (where the sub-matrices are replaced by $\widehat{\mathbf{B}}_{0,0}, \widehat{\mathbf{B}}_{0,1}, \widehat{\mathbf{B}}_{1,0}, \widehat{\mathbf{B}}_{1,1}, \widehat{\mathbf{A}}_0, \widehat{\mathbf{A}}_1$ and $\widehat{\mathbf{A}}_2$, respectively). The sub-matrices are calculated as follows:

$$\widehat{\mathbf{B}}_{i,j} = \begin{cases} \mathbf{I} + \frac{\mathbf{B}_{i,j}}{\lambda}, & i = j, \\ \frac{\mathbf{B}_{i,j}}{\lambda}, & i \neq j, \end{cases} \quad \text{and} \quad \widehat{\mathbf{A}}_i = \begin{cases} \mathbf{I} + \frac{\mathbf{A}_i}{\lambda}, & i = 1, \\ \frac{\mathbf{A}_i}{\lambda}, & i \neq 1. \end{cases}$$

Let $\mathbf{U}^{(k)}$ be the state probability distribution matrix after k epochs in the DTMC with transition matrix $\mathbf{P}$. That is, entry (i,j) of $\mathbf{U}^{(k)}$ is the probability that j is reached from i in k steps. $\mathbf{U}^{(k)}$ can be derived recursively as:

$$\mathbf{U}^{(0)} = \mathbf{I}, \quad \text{and} \quad \mathbf{U}^{(k)} = \mathbf{U}^{(k-1)}\mathbf{P}, \quad k \in \mathbb{N}^+. \tag{3.7}$$

Then, the matrix of transient state probabilities for the original CTMC at time t, can be calculated as:

$$\mathbf{V}(t) = \sum_{k=0}^{\infty} \psi(\lambda t; k)\mathbf{P}^k = \sum_{k=0}^{\infty} \psi(\lambda t; k)\mathbf{U}^{(k)}, \tag{3.8}$$

where $\psi(\lambda t; k)$ is the probability of k events occurring in the interval $[0, t)$ in a Poisson process with rate λ. The probability distribution in the DTMC after k steps is described by $\mathbf{V}(0) \cdot \mathbf{P}^k$ (note that $\mathbf{V}(0) = \mathbf{I}$).

To avoid the infinite summation over the number of steps k, the sum (3.8) needs to be truncated. We denote the approximation of $\mathbf{V}(t)$ that has been calculated with up to n terms of the summation with $\mathbf{V}^{(n)}(t)$:

$$\mathbf{V}^{(n)}(t) = \sum_{k=0}^{n} \psi(\lambda t; k)\mathbf{U}^{(k)}. \tag{3.9}$$

We can compute $\mathbf{V}^{(n+1)}(t)$ as:

$$\mathbf{V}^{(n+1)}(t) = \mathbf{V}^{(n)}(t) + \psi(\lambda t; n+1)\mathbf{U}^{(n+1)}. \tag{3.10}$$

Note that $\mathbf{V}^{(n)}(t)$ follows the structure of the previous $\mathbf{U}^{(m)}$ $(m \leq n)$ in terms of zeroes and non-zeroes because any non-zero entry in $\mathbf{V}^{(n)}$ corresponds to a non-zero entry in at least one $\mathbf{U}^{(m)}(m \leq n)$. We denote a maximum bound on the error that possibly occurs in an entry of $\mathbf{V}(t)$ when the series is truncated after n steps as $\varepsilon_{t,\lambda}^{(n)}$. We have

$$\left\| \sum_{k=n+1}^{\infty} \psi(\lambda t; k)\mathbf{U}^{(k)} \right\| \leq 1 - \sum_{k=0}^{n} e^{-\lambda t} \frac{(\lambda t)^k}{k!} = \varepsilon_{t,\lambda}^{(n)}. \tag{3.11}$$

Note that for given error $\varepsilon_{t,\lambda}^{(n)}$, λ and t, n can be computed *a priori*, cf. [36, 38]. For a given number of steps n, $\varepsilon_{t,\lambda}^{(n)}$ increases linearly with $\lambda \cdot t$ and decreases linearly with n.

Finally, observe that the matrices $\mathbf{V}^{(n)}(t)$ and $\mathbf{U}^{(n)}$ are of infinite size. However, exploiting the repetitive structure of QBDs and the truncation given by uniformization we can give a finite representation that depends on the number of considered steps n for a given error bound, as will be presented next.

3.3.2 Finite representation

From every single state, only a finite number of states is reachable in n steps. The transient probability computed by uniformization to reach one of the non-reachable states is zero. Hence, for a single starting state it is sufficient to consider only the finite set of reachable states. This was already observed in [34, 57, 84]. When simultaneously considering every state of the infinite state space as starting state, one would have to consider an infinite number of finite parts of the QBD, which is practically not feasible. However, given a finite number n of steps, there is a repeating level l from which onwards the boundary level cannot be reached anymore. Therefore, the finite part of the QBD that needs to be considered for starting states from repeating levels l onward does not contain states of the boundary level. The structure of all these finite parts is identical, only shifted appropriately, due to the regular structure of the QBD. This implies that we obtain identical

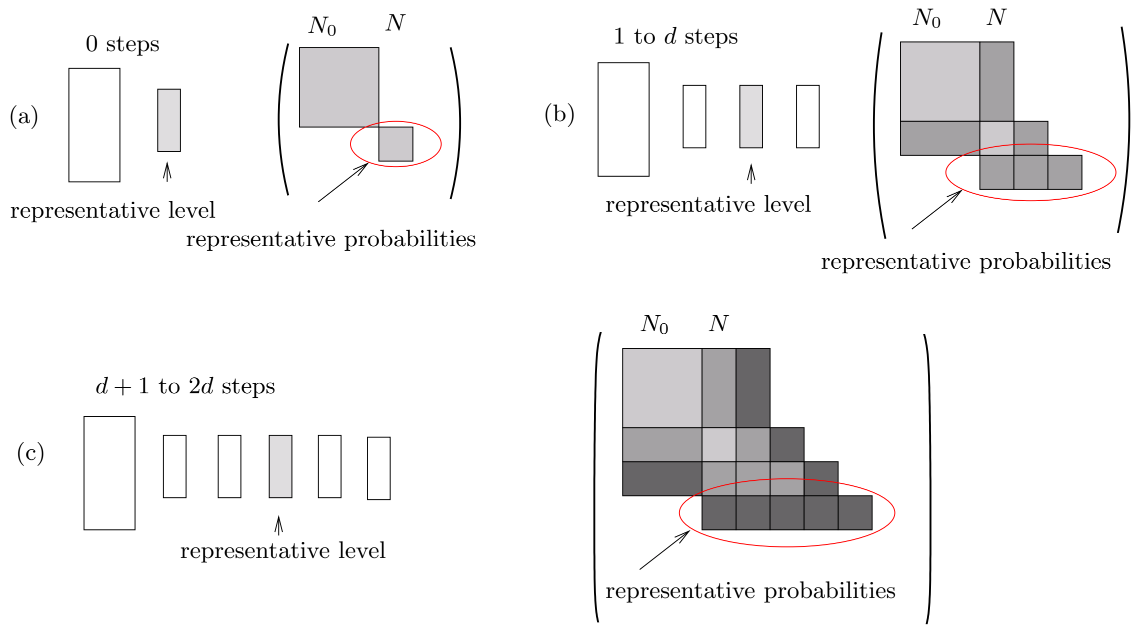


Figure 3.4: Considered part of the state space (left) and finite representation of $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$ (right), depending on the number of considered steps

transient probabilities (shifted appropriately) for corresponding states in repeating levels numbered at least l, within the error bounds of uniformization, given n steps. Therefore, we can use the states of level l as *representatives* for all corresponding states at higher levels. In fact, we restrict the computation to a finite number of starting states and still perform a comprehensive transient analysis *for every possible state as starting state.*

For a finite representation of the matrices $\mathbf{V}^{(n)}(t)$ and $\mathbf{U}^{(n)}$, it is now sufficient to store all non-zero entries for starting states of levels up to l. The size of the finite representation depends on the considered number of steps n, hence, on the time, the uniformization rate, and the required accuracy.

3.3.3 Probability distribution after n epochs

We now address the growth of the matrices $\mathbf{U}^{(n)}$ in the course of the computation. Figure 3.4(a) shows that the dimension of the finite representation of $\mathbf{U}^{(0)}$ is $(N_0 + N) \times (N_0 + N)$. Since $n = 0$, we cannot leave a level and the first repeating level is already representative for all (other) repeating levels. In case $n = 1$, we can reach the next higher or the next lower level. Since the next lower level is the boundary level, the first repeating level cannot be used as representative, but we can use the second repeating level as representative, as shown in Figure 3.4(b). Since $n = 1$, it is possible to reach the next higher level as well; thus we have to consider starting in one of the first three levels (including the boundary level) and ending up in one of the

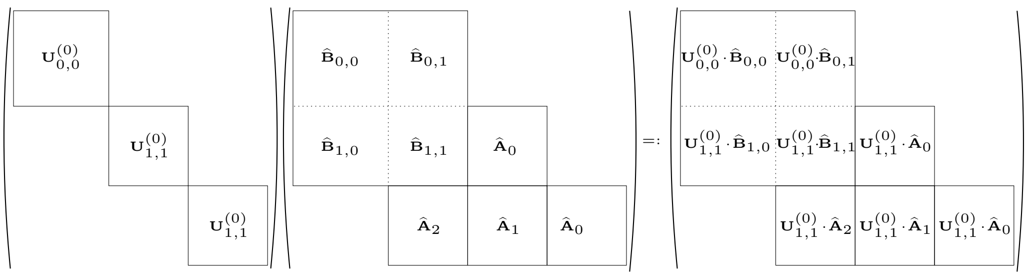

$\mathbf{U}^{(0)}_{0,0}$
$\mathbf{U}^{(0)}_{1,1}$
$\mathbf{U}^{(0)}_{1,1}$
$\hat{\mathbf{B}}_{0,0}$
$\hat{\mathbf{B}}_{0,1}$
$\hat{\mathbf{B}}_{1,0}$
$\hat{\mathbf{B}}_{1,1}$
$\hat{\mathbf{A}}_0$
$\hat{\mathbf{A}}_2$
$\hat{\mathbf{A}}_1$
$\hat{\mathbf{A}}_0$
$=:$
$\mathbf{U}^{(0)}_{0,0} \cdot \hat{\mathbf{B}}_{0,0}$
$\mathbf{U}^{(0)}_{0,0} \cdot \hat{\mathbf{B}}_{0,1}$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{B}}_{1,0}$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{B}}_{1,1}$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{A}}_0$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{A}}_2$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{A}}_1$
$\mathbf{U}^{(0)}_{1,1} \cdot \hat{\mathbf{A}}_0$

Figure 3.5: Computation of $\mathbf{U}^{(0)} \cdot \mathbf{P} = \mathbf{U}^{(1)}$

first four levels. The dimension of the finite representation of $\mathbf{U}^{(1)}$ therefore needs to be $(N_0 + 2N) \times (N_0 + 3N)$. With a symmetric level diameter d, we will need at least another $d - 1$ steps before possibly reaching the next higher repeating level. Thus, the size of all $\mathbf{U}^{(n)}$, for $n = 1, \cdots, d$, will be the same as for $n = 1$. Figure 3.4(c) shows the finite representation of matrices $\mathbf{U}^{(n)}$, for $n = d + 1, \cdots, 2d$. From a given level, we can reach at most the next two higher or lower levels. Therefore, we have to pick a new representative: the third repeating level. Starting from this representative, we can reach the next two higher repeating levels. We have to attach another row (of blocks of states) for the new representative, and in every other row we have to attach one block of states to the left (the next lower) and one to the right (the next higher level), wherever possible. The dimension of the finite representation is then $(N_0 + 3N) \times (N_0 + 5N)$, for all $\mathbf{U}^{(n)}$, for $n = d+1, \cdots, 2d$. In general, for a given number of steps $n \geq 1$ and level diameter d, the maximum number l of levels reachable from a representative level in one direction (up or down) is given by

$$l = ((n - 1) \text{ div } d) + 1. \tag{3.12}$$

The size of the matrix $\mathbf{U}^{(n)}$ is then determined by l. The dimension of its finite representation is $(N_0+(l+1)N)\times(N_0+(2l+1)N)$. As before, the finite representation of the matrix $\mathbf{V}^{(n)}$ has the same dimension.

3.3.4 Uniformization with representatives

We now proceed with the actual computation of $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$ according to Equation (3.10). Starting with $n = 0$, and thus with a small finite portion of the QBD, cf. Figure 3.4(a), we increase n step by step, thus increasing accuracy and size of the considered finite representation of the QBD. However, in each iteration we always use the smallest possible representation. The matrices $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$ have a block structure, according to the levels of a QBD; we denote the blocks that give the probabilities from states in level i to states in level j as $\mathbf{U}^{(n)}_{i,j}$ and $\mathbf{V}_{i,j}(t)$.

Starting with $\mathbf{U}^{(0)}$ (with dimension of the finite representation $(N_0 + N) \times (N_0 + N)$), the computation of $\mathbf{U}^{(1)}$ is visualized in Figure 3.5: we multiply the finite representation of $\mathbf{U}^{(0)}$, where one row of blocks is added for the new representative repeating level, with a finite portion of $\mathbf{P}$ that consists of three block rows (for the three considered starting levels) and of four block columns (for the four levels that can be reached). In general, for $n \geq 1, \mathbf{U}^{(n)}$ is computed as $\mathbf{U}^{(n-1)} \cdot \mathbf{P}$, cf. (3.7), as follows:

$$\begin{aligned}
\mathbf{U}_{i,0}^{(n)} &= \mathbf{U}_{i,0}^{(n-1)} \cdot \widehat{\mathbf{B}}_{0,0} + \mathbf{U}_{i,1}^{(n-1)} \cdot \widehat{\mathbf{B}}_{1,0}, && \text{for } i = 0, \cdots, l,\\
\mathbf{U}_{i,1}^{(n)} &= \mathbf{U}_{i,0}^{(n-1)} \cdot \widehat{\mathbf{B}}_{0,1} + \mathbf{U}_{i,1}^{(n-1)} \cdot \widehat{\mathbf{B}}_{1,1} + \mathbf{U}_{i,2}^{(n-1)} \cdot \widehat{\mathbf{A}}_2, && \text{for } i = 0, \cdots, l+1,\\
\mathbf{U}_{i,j}^{(n)} &= \mathbf{U}_{i,j-1}^{(n-1)} \cdot \widehat{\mathbf{A}}_0 + \mathbf{U}_{i,j}^{(n-1)} \cdot \widehat{\mathbf{A}}_1 + \mathbf{U}_{i,j+1}^{(n-1)} \cdot \widehat{\mathbf{A}}_2, && \text{for } i = 0, \cdots, l+1,\\
& && \quad j = 2, \cdots, i+l,
\end{aligned} \tag{3.13}$$

where l is computed as in (3.12).

Proposition 7 (Corresponding blocks). For a given number of steps n and for all $i > n$ and for all $j \geq n + i$, the blocks $\mathbf{U}_{i,j}^{(n)}$ contain the same probabilities as the corresponding representative block $\mathbf{U}_{n,j-i+n}^{(n)}$. Hence, our finite representation is correct. □

Proof (by induction on n). (Basis $n = 0$) We know from Equation (3.7) that $\mathbf{U}^{(0)}$ is initialized with the identity matrix:

$$\mathbf{U}_{i,j}^{(0)} = \begin{cases} \mathbf{I}, & i = j, \\ 0, & \text{otherwise,} \end{cases} \quad \text{for } i, j > 0.$$

This equals the above proposition because for $i = j$, it follows $\mathbf{U}_{i,j}^{(0)} = \mathbf{I} = \mathbf{U}_{0,j-i}^{(0)} = \mathbf{U}_{0,0}^{(0)}$, and for $i \neq j$ it follows $\mathbf{U}_{i,j}^{(0)} = 0 = \mathbf{U}_{0,j-i}^{(0)}$. Hence, for $n = 0$, the above proposition is true.

(Induction step) The induction hypothesis (IH) is that, for $k \geq 0$,

$$\mathbf{U}_{i,j}^{(k)} = \mathbf{U}_{k,j-i+k}^{(k)} \text{ for } i > k \text{ and for all } j \geq k + i.$$

We know from (3.13) that for $j > k$ and for $i \geq k + j$

$$\begin{aligned}
\mathbf{U}_{i,j}^{(k+1)} &= \mathbf{U}_{i,j-1}^{(k)} \cdot \widehat{\mathbf{A}}_0 + \mathbf{U}_{i,j}^{(k)} \cdot \widehat{\mathbf{A}}_1 + \mathbf{U}_{i,j+1}^{(k)} \cdot \widehat{\mathbf{A}}_2\\
&\overset{\text{IH}}{=} \mathbf{U}_{k,j-1-i+k}^{(k)} \cdot \widehat{\mathbf{A}}_0 + \mathbf{U}_{k,j-i+k}^{(k)} \cdot \widehat{\mathbf{A}}_1 + \mathbf{U}_{k,j+1-i+k}^{(k)} \cdot \widehat{\mathbf{A}}_2\\
&\overset{(3.13)}{=} \mathbf{U}_{k,j-i+k}^{(k+1)},
\end{aligned} \tag{3.14}$$

which proves the proposition for $n = k + 1$. □

Due to the block structure of $\mathbf{V}^{(n)}(t)$, we can rewrite (3.9) as:

$$\mathbf{V}_{i,j}^{(n)}(t) = \mathbf{V}_{i,j}^{(n-1)}(t) + \psi(\lambda t; n) \cdot \mathbf{U}_{i,j}^{(n)}, \tag{3.15}$$

for $i = 0, \cdots, l+1$, and for $j = \max\{0, i-l\}, \cdots, i+l$. After d steps, the size of $\mathbf{V}^{(n)}(t)$ will have to be adapted. This comes at no computational costs since the block matrices that need to be appended are either zero, or just copies of block matrices that have been computed already.

3.3.5 Complexity

The level index l_k of the representative increases with the number of considered steps k and decreases with the symmetric level diameter d. In the k-th iteration, we actually consider the states of the boundary level and of $l_k + 1$ repeating levels as starting states, and the states of the boundary level and of $2l_k + 1$ repeating levels as end states. The boundary level has N_0 states and each repeating level has N states, resulting in matrices with $(N_0 + (l_k + 1)N) \times (N_0 + (2l_k + 1)N)$ entries, so that the storage requirement grows with the level index of the representative. If n is the maximum number of steps considered, the overall storage complexity for the three probability matrices $\mathbf{U}^{(n-1)}$, $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}$ is $\mathcal{O}(3(N_0^2 + l_n N_0 N + l_n^2 N^2))$.

Let ν denote the average number of transitions originating from a single state in the QBD. Assuming a sparse representation, the discrete transition matrix $\mathbf{P}$ has storage complexity $\mathcal{O}(\nu(N_0 + 2N))$. In the k-th iteration, the multiplication of matrix $\mathbf{U}^{(n-1)}$ with $\mathbf{P}$ is carried out in $\mathcal{O}(\nu(N_0 + (l_k + 1)N))$ time. For n the maximum number of considered steps, the overall time complexity therefore equals $\mathcal{O}(n \cdot \nu(N_0 + (\frac{l_n}{2} + 1)N))$.

Note that the iteration costs per level increase. However, when probability matrices of the size $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}$ would be used throughout the complete computation, the iteration costs would be much higher.

3.4 Different time intervals for the until operator

In this section we present the algorithms to compute the satisfaction set for CSL formulas with the until operator for the different time intervals. Note that the algorithms, as presented in the following, do not stop in case the computed probability for at least one starting state equals the probability bound p. The presented iterations will then never stop; however, this is highly unlikely to occur in practice.

An efficient algorithm for the time-bounded until operator with time interval $I = [0, t]$ will be discussed in 3.4.1 and the corresponding pseudocode is provided in Section 3.4.2. Section 3.4.3 presents the algorithm for model checking the time-bounded until operator with time interval $I = [t_1, t_2]$. The algorithm for the time-unbounded until operator is discussed in Section 3.4.4 for the time interval $I = [0, \infty)$ and in Section 3.4.5 for the time interval $I = [t, \infty)$. Furthermore, we provide a proof of Proposition 5 for the until operator in Section 3.4.6.

3.4.1 The simplest case $I = [0, t]$

As for any uniformization-based technique, we can compute the number of steps that needs to be taken into account *a priori* for a given error bound $\varepsilon_{t,\lambda}$. However, this may introduce several problems in a model checking context. First of all, such a statically computed number of steps might be larger than is really needed to determine whether a CSL property is satisfied or not. Furthermore, in other cases the preset accuracy (hence, number of steps) might not be sufficient to decide whether the computed probability meets the required bound. To overcome both problems, we propose a dynamic termination criterion, which we claim to be optimally efficient in the current setting.

For model checking an until-formula $\mathcal{P}_{\bowtie p}(\Phi\,\mathcal{U}^{[0,t]}\Psi)$ we have to compare for each starting state the probability to take a $(\Phi\,\mathcal{U}^{[0,t]}\Psi)$-path with the probability bound p. In the transformed QBD $\mathcal{Q}[\neg\Phi \vee \Psi]$ the set of goal states consists of all Ψ-states. We denote the probability to end up in a Ψ-state, given starting state s, as $\gamma_{\mathsf{s}}(t)$. For the time interval $I = [0, t]$, we have:

$$\gamma_{\mathsf{s}}(t) = \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}, \mathsf{s}', t). \tag{3.16}$$

Note that the vector $\gamma(t)$ consists of sub-vectors corresponding to the levels of the QBD that are considered with uniformization. The approximation of $\gamma(t)$ after n iterations is called $\gamma^{(n)}(t) = \mathbf{V}^{(n)}(t) \cdot \gamma(0)$, and

$$\gamma_s(0) = \begin{cases} 1, & \mathsf{s} \models \Psi, \\ 0, & \text{otherwise.} \end{cases}$$

In principle, $\gamma^{(n)}(t)$ is of infinite size, but we can cut it to a finite representation, as from a representative level on, all levels contain the same values. It is also possible to derive Equation (3.13) directly for $\gamma(t)$ and then use this vector for the computation. When increasing the number of considered steps n, the entries of $\gamma^{(n)}(t)$ increase monotonously. Thus, comparing entries of the probability vector $\gamma^{(n)}(t)$ with the bound p on a regular basis, we might be able to decide whether the probability meets the bound p after a smaller number of iterations than after the number of iterations that is computed a priori.

With uniformization with representatives, the computed approximation after n steps always underestimates the actual probability. Recall that $\varepsilon_{t,\lambda}^{(n)}$ is the maximum error of uniformization after n iteration steps (cf. (3.11)), such that $\gamma_{\mathsf{s}}(t) \leq \gamma_{\mathsf{s}}^{(n)}(t) + \varepsilon_{t,\lambda}^{(n)}$ for time interval $I = [0, t]$. From (3.11) it follows that the value of $\varepsilon_{t,\lambda}^{(n)}$ decreases as n increases. Exploiting the above inequality, we obtain the following termination criteria:

$$\begin{array}{lrcl} \text{(a)} & \gamma_{\mathsf{s}}^{(n)}(t) \geq p & \Rightarrow & \gamma_{\mathsf{s}}(t) \geq p, \\ \text{(b)} & \gamma_{\mathsf{s}}^{(n)}(t) < p - \varepsilon_{t,\lambda}^{(n)} & \Rightarrow & \gamma_{\mathsf{s}}(t) < p. \end{array}$$

These criteria can be exploited as follows. Starting with a small number of steps, we check whether for the current approximation one of the inequalities (a) or (b) holds for all starting states. If this is not the case we continue, check again, etc., until either of the termination criteria holds. However, if for one of the starting states $\mathsf{s} \in \mathsf{S}$ we have $\gamma_{\mathsf{s}}(t) = p$, the iteration never stops, as neither of the termination criteria ever holds. Given $\neg\Phi \vee \Psi$ is independent as of level l and the iteration stops after n steps for all considered starting states, then level $l+n$ is representative for all levels larger $l+n$.

Hence, we used an approximate algorithm to construct an exact decision procedure for model checking the until operator. The approximate algorithm, uniformization, computes the transient probabilities for a given error bound. Comparing the computed probability with the given probability bound and the uniformization error allows us to decide for a given starting state whether or not a CSL formula that contains the until operator is valid. Note that this approach could also be used for model checking finite CTMCs. In [8] the truncation error due to uniformization is not taken into account.

Recall, that the point interval until can be computed just like the time bounded until, with the only difference that the transient probabilities have to be computed on $\mathcal{Q}[\neg\Phi]$ and that the goal states s' have to fulfill $\Phi \wedge \Psi$.

3.4.2 Pseudocode algorithm for $I = [0, t]$

We present the algorithms needed for model checking $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]}\, \Psi)$ in pseudocode.

Algorithm 6 uniformize(p, t, k, $\gamma(0)$) : $\gamma(t)$, k

```
start = $\bigcup_{j=0}^{k} S^j$;                    (* set of starting states *)
n = 0;
$\mathbf{U}^{(0)} = \mathbf{I}$;
$\mathbf{V}^{(0)} = \mathbf{I}$;
$\gamma^{(0)}(t) = \gamma(0) \cdot \mathbf{V}^{(0)}$;
set $\varepsilon_{t,\lambda}^{(0)}$;                 (* according to Eq. (3.11) *)
while (¬ check($\gamma^{(n)}(t)$ ,$\varepsilon_{t,\lambda}^{(n)}$, $\bowtie p$) ) do
  n = n+1;
  start = start ∪ $S^{k+n}$;
  update $\mathbf{U}^{(n)}$;                         (* according to Eq. (3.13) *)
  update $\mathbf{V}^{(n)}$;                         (* according to Eq. (3.15) *)
  update $\varepsilon_{t,\lambda}^{(n)}$;            (* according to Eq. (3.11) *)
  $\gamma^{(n)}(t) = \gamma(0) \cdot \mathbf{V}^{(n)}$;
end while
k = k + n;
return $\gamma^{(n)}(t)$, k;
```

Algorithm 7 check($\gamma^{(n)}(t), \varepsilon^{(n)}_{t,\lambda}, \bowtie \mathtt{p}$) : boolean

```
finished $= 1$;
for all $s \in S$ do
  if $\bowtie \in \{<, \geq\}$ then
    if $\neg$ ( $\gamma^{(n)}_s(t) \geq \mathtt{p}$) then
      if $\neg(\gamma^{(n)}_s(t) < \mathtt{p} - \varepsilon^{(n)}_{t,\lambda})$ then
        finished $= 0$;                    (* not enough iterations yet *)
      end if
    end if
  else if $\bowtie \in \{>, \leq\}$ then
    if $\neg$ ( $\gamma^{(n)}_s(t) > \mathtt{p}$) then
      if $\neg(\gamma^{(n)}_s(t) \leq \mathtt{p} - \varepsilon^{(n)}_{t,\lambda})$ then
        finished $= 0$;                    (* not enough iterations yet *)
      end if
    end if
  end if
end for
return (finished);
```

Algorithm 8 $Sat^{[0,t]}_{\mathcal{U}}(\bowtie p, t, \Phi_1, \Phi_2)$: $\bigcup_{i=0}^{k} Sat^i$

```
begin
$\Phi_1$ independent as of $\mathtt{k}_1$;
$\Phi_2$ independent as of $\mathtt{k}_2$;
$\mathtt{k} = \max(\mathtt{k}_1, \mathtt{k}_2)$;
make absorbing($\neg\Phi_1 \vee \Phi_2$);
set $\gamma(0)$;                           (* according to Eq. (3.16) *)
$\gamma(t)$, $\mathtt{k}$ = uniformize($\mathtt{p}$, $\mathtt{t}$ , $\mathtt{k}$, $\gamma(0)$);
for all $i \in \{0, \ldots, \mathtt{k}\}$ do
  for all $s \in S^i$ do
    if satisfy$_{\mathcal{U}}(\mathtt{s}, \gamma_s(t), \bowtie \mathtt{p})$ then
      $Sat^i = Sat^i \cup \{s\}$;
    end if
  end for
end for
return $\bigcup_{i=0}^{k} Sat^i$;
end
```

Algorithm 6 presents uniformization employing the dynamic termination criterion in pseudocode. The input parameters of the algorithm are the probability bound p, the time t, the level independence of the inner formula $\neg\Phi \vee \Psi$, and the vector $\gamma(0)$ that indicates for all states whether s is a goal state or not. Algorithm 6 returns a vector of probabilities $\gamma(t)$ to end in a goal state, the error after n iterations $\varepsilon_{t,\lambda}^{(n)}$, and the level k that contains the representative probabilities.

The termination criterion is coded in Algorithm 7 that returns `FALSE` if the current number of iterations is not yet enough to decide the validity of the until formula for all starting states. In that case Algorithm 6 enters another iteration.

In Algorithm 8 model checking $\mathcal{P}_{\bowtie p}(\Phi_1\ \mathcal{U}^{[0,t]}\Phi_2)$ is presented in pseudocode. To compute the transient probabilities, Algorithm 6 is called and Algorithm 9 then returns whether or not a given starting state fulfills $\mathcal{P}_{\bowtie p}(\Phi_1\ \mathcal{U}^{[0,t]}\Phi_2)$ for the different comparison operators $\bowtie\, \in \{<, >, \leq, \geq\}$.

Algorithm 9 satisfy$_{\mathcal{U}}(\mathsf{s}, \gamma_s(t), \bowtie \mathsf{p})$: boolean

```
if ($\bowtie$ is $<$) then
  return ($\neg(\gamma_s(t) \geq p)$);
else if ($\bowtie$ is $>$) then
  return ($\gamma_s(t) > p$);
else if ($\bowtie$ is $\leq$) then
  return ($\neg(\gamma_s(t) > p)$);
else if ($\bowtie$ is $\geq$) then
  return ($\gamma_s(t) \geq p$);
end if
```

3.4.3 The bounded case $I = [t_1, t_2]$

As presented in Section 3.2.7, the computation of the transient probabilities for the time interval $I = [t_1, t_2]$, is split into two parts. We need to combine the transient probabilities in the transformed QBD $\mathcal{Q}[\neg\Phi]$ at time t_1 with the transient probabilities in $\mathcal{Q}[\neg\Phi \vee \Psi]$ at time $t_2 - t_1$, as follows:

$$\begin{aligned}
\gamma_{\mathsf{s}}(t_2) &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^j(\Psi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1) \cdot \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1) \\
&= \underbrace{\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Phi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1)} \cdot \underbrace{\sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^j(\Psi)} \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1)} \\
&= \qquad\qquad \gamma_{\mathsf{s}}(t_1) \qquad\qquad \cdot \qquad\qquad \gamma_{\mathsf{s}'}(t_2 - t_1)
\end{aligned}$$

The first equation is rewritten by separating the transient probabilities of the two parts. For a given a priori error $\varepsilon_{t_1,\lambda_1}$ an initial distribution $\gamma_{\mathsf{s}}^{(n_1)}(t_1)$ for the first part of the transient probabilities is computed with uniformization in n_1 steps. The second part of the transient probabilities is computed with uniformization with representatives. The approximation of $\gamma_{\mathsf{s}'}(t_2-t_1)$ with n_2 steps is denoted as $\gamma_{\mathsf{s}'}^{(n_2)}(t_2-t_1)$ and induces the error $\varepsilon_{t_2-t_1,\lambda_2}^{(n_2)}$. Both approximations are an underestimation of the actual probability:

$$\gamma_{\mathsf{s}}(t_2) \leq \left(\gamma_{\mathsf{s}}^{(n_1)}(t_1) + \varepsilon_{t_1,\lambda_1}^{(n_1)}\right) \cdot \left(\gamma_{\mathsf{s}'}^{(n_2)}(t_2 - t_1) + \varepsilon_{t_2-t_1,\lambda_2}^{(n_2)}\right). \tag{3.17}$$

For a given initial distribution $\gamma_{\mathsf{s}}^{(n_1)}(t_1)$, we iterate n_2 until for every considered starting state one of the following termination criteria holds:

$$\begin{array}{lrcl}
\text{(a)} & \gamma_{\mathsf{s}}^{(n_1)}(t_1) \cdot \gamma_{\mathsf{s}'}^{(n_2)}(t_2 - t_1) > p & \Rightarrow & \gamma_{\mathsf{s}}(t_2) > p \\
\text{(b)} & \gamma_{\mathsf{s}'}^{(n_2)}(t_2 - t_1) < \frac{p}{\gamma_{\mathsf{s}}^{(n_1)} + \varepsilon_{t_1,\lambda_1}^{(n_1)}} - \varepsilon_{t_2-t_1,\lambda_2}^{(n_2)} & \Rightarrow & \gamma_{\mathsf{s}}(t_2) < p.
\end{array}$$

Given $\neg\Phi \vee \Psi$ is independent as of level l, we have to find n_2 such that either (a) or (b) is fulfilled for every considered starting state. Given that the initial distribution is computed with an a priori known number of steps n_1, level $l + n_1 + n_2$ serves as representative for all levels $i > l + n_1 + n_2$.

3.4.4 The unbounded case $I = [0, \infty)$

In Section 3.2.7 we showed how the model checking of an unbounded until operator relies on the computation of the steady-state probabilities in the absorbing QBD $\mathcal{Q}[\neg\Phi \vee \Psi]$. Recall that $\mathcal{Q}[\neg\Phi \vee \Psi]$ may not be strongly connected. However, the steady-state probability of the set of absorbing Ψ-states is independent of the residence time in each state but only depends on the branching probabilities and the starting state. We can therefore switch to the discrete-time Markov chain that is derived from the embedded discrete-time Markov that corresponds to $\mathcal{Q}[\neg\Phi \vee \Psi]$, by removing self-loops. This probability matrix is denoted $\mathbf{H}$ where

$$\mathbf{H}(\mathsf{s}, \mathsf{s}') = \frac{\mathbf{Q}(\mathsf{s}, \mathsf{s}')}{-\mathbf{Q}(\mathsf{s}, \mathsf{s})} \text{ for } \mathsf{s} \neq \mathsf{s}', \text{ and } \mathbf{H}(\mathsf{s}, \mathsf{s}) = 0.$$

Note that $\mathbf{H}^n(\mathsf{s},\mathsf{s}')$ denotes the probability to reach s' from s in exactly step n. If the level of the starting state is l, the desired steady-state probability is

$$\begin{aligned}\pi(\mathsf{s}, Sat(\Psi)) &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \pi(\mathsf{s},\mathsf{s}') \\ &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s},\mathsf{s}') \\ &= \sum_{n=0}^{\infty} \sum_{i=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}').\end{aligned}$$

The second equality replaces the steady-state probability $\pi(\mathsf{s},\mathsf{s}')$ by the probability to reach a Ψ state s' in zero up to infinitely many steps, i.e., by $\sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s},\mathsf{s}')$. The third equality follows from the fact that only a finite number of levels is reachable in n steps. Using this equation, representative probabilities can be computed for an increasing number of considered steps n, as is the case for uniformization. The termination criterion is adopted from Section 3.2.5. In every step we have to compare the approximations computed with N steps:

$$\tilde{\pi}^{(N)}(\mathsf{s}, Sat(\Psi)) = \sum_{n=0}^{N} \sum_{i=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}' \in Sat^i(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}')$$

and $\tilde{\pi}^{(N)}(\mathsf{s}, Sat(\neg\Psi))$, with p and $1-p$, respectively, until we can stop the iteration. As soon as one of the left-hand side inequalities becomes true, we can stop:

$$\begin{array}{lrcl}(a) & \tilde{\pi}^{(N)}(Sat(\Phi)) > p & \Rightarrow & \pi(Sat(\Phi)) > p, \\ (b) & \tilde{\pi}^{(N)}(Sat(\neg\Phi)) > 1-p & \Rightarrow & \pi(Sat(\Phi)) < p.\end{array}$$

Given Ψ is independent as of level l, we have to find N, such that either (a) or (b) holds for all considered starting states. Level $l+N$ is then the representative level for all levels larger $l+N$, as the structure of the QBD remains the same for N steps to the left and for N steps to the right.

3.4.5 The unbounded case $I = [t, \infty)$

As presented in Section 3.2.7, model checking the time bounded until operator for the time interval $I = [t, \infty)$, consists of two parts. We need to combine the transient probabilities in the transformed QBD $\mathcal{Q}[\neg\Phi]$ at time t with the steady-state probabilities in $\mathcal{Q}[\neg\Phi \vee \Psi]$. If the level of the starting state is l, the desired steady-state probability is:

$$\begin{aligned}
&\pi(s, Sat(\Phi))\\
=&\sum_{i=0}^{\infty}\sum_{\mathsf{s}'\in Sat^i(\Phi)}\sum_{j=0}^{\infty}\sum_{\mathsf{s}''\in Sat^j(\Psi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s},\mathsf{s}',t)\cdot \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}',\mathsf{s}'')\\
=&\sum_{i=0}^{\infty}\sum_{\mathsf{s}'\in Sat^i(\Phi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s},\mathsf{s}',t)\cdot \sum_{j=0}^{\infty}\sum_{\mathsf{s}''\in Sat^j(\Psi)} \pi^{\mathcal{Q}[\neg\Phi\vee\Psi]}(\mathsf{s}',\mathsf{s}'')\\
=&\sum_{i=0}^{\infty}\sum_{\mathsf{s}'\in Sat^i(\Phi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s},\mathsf{s}',t)\cdot \sum_{j=0}^{\infty}\sum_{\mathsf{s}''\in Sat^j(\Psi)}\sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s}',\mathsf{s}'')\\
=&\sum_{i=0}^{\infty}\sum_{\mathsf{s}'\in Sat^i(\Phi)} \pi^{\mathcal{Q}[\neg\Phi]}(\mathsf{s},\mathsf{s}',t)\cdot \sum_{n=0}^{\infty}\sum_{j=\max(0,l-n)}^{l+n}\sum_{\mathsf{s}''\in Sat^j(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}').
\end{aligned}$$

The first equation can be rewritten by separating the transient from the steady-state probabilities. The steady-state probabilities are expressed via the embedded discrete-time QBD with probability matrix $\mathbf{H}$, as presented in Section 3.4.4.

To actually compute $\pi(s, Sat(\Phi))$, an initial distribution for the steady-state probabilities, denoted $\widetilde{\pi}_s(t)$ is computed for a given error ε. The steady-state probabilities are approximated as follows:

$$\widetilde{\pi}^{(N)}(s', Sat(\Phi)) = \sum_{n=0}^{N}\sum_{j=\max(0,l-n)}^{l+n}\sum_{\mathsf{s}''\in Sat^j(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}').$$

Combining the initial distribution with the approximated steady-state probabilities $\widetilde{\pi}^N(s', Sat(\Phi))$ for increasing N, we compare with p and combining the initial distribution with $\widetilde{\pi}^N(s', Sat(\neg\Phi))$ we compare with $1-p$ for every N, until we can stop the iteration. As soon as some of the left-hand side inequalities becomes true, we can stop:

$$\text{(a)} \sum_{s'\in Sat(\Phi)} \widetilde{\pi}(s,s',t)\cdot\widetilde{\pi}^{(N)}(s', Sat(\Phi)) > p \qquad \Rightarrow \qquad \pi(s, Sat(\Phi)) > p$$

$$\text{(b)} \sum_{s'\in Sat(\Phi)} \widetilde{\pi}(s,s',t)\cdot\widetilde{\pi}^{(N)}(s', Sat(\neg\Phi)) > 1-p \qquad \Rightarrow \qquad \pi(s, Sat(\Phi)) < p$$

Note that the a priori error that is chosen for the initial distribution does not influence the above inequalities as the computed probability is always an underestimation. Thus, in case the underestimation is already greater than p or $1-p$, we can be sure that the actual probability is so as well. The a priori error of the initial distribution should be chosen reasonably small, in order to take enough probability mass into account, otherwise the iteration might not stop.

Given $\neg\Phi \vee \Psi$ is independent as of level l and the number of steps considered by uniformization for an a priori error $\varepsilon_{\lambda,t}$ is n. Then, in case the above iteration stops after N steps for all starting states $\mathsf{s} \in \bigcup_{i=0}^{n+N} S^i$, level $l+n+N$ is representative for all levels $i > l+n+N$.

3.4.6 Proof of Proposition 5 for the Until operator

After having presented the algorithms for the model checking of until formulas with different time intervals we now discuss the justification of Proposition 5. We want to show that any until formula $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I\, \Psi)$ is independent as of level k for some finite k, under the assumption that for no state s the probability measure is exactly equal to p, hence, $Prob(\mathsf{s}, \Phi\, \mathcal{U}^I\, \Psi) \neq p$.

We do this for time-bounded until formulas $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]}\, \Psi)$ in more detail. Assume that Φ is level independent as of k_1 and Ψ is level independent as of k_2. The maximum of k_1 and k_2 is then denoted k'. The termination criterion (cf. Section 3.4.1) of uniformization with representatives ensures that for every state $\mathsf{s} = (i, k')$ with $i \in \{0, \ldots, N-1\}$ there exists an $N_i \geq 1$ such that either

$$\forall l \geq N_i : \gamma_{(i,l)}(t) > p \text{ or } \forall l \geq N_i : \gamma_{(i,l)}(t) < p.$$

The maximum of all N_i is then the index k for level independence of the formula $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]}\, \Psi)$.

However, we still have to discuss that the algorithm always terminates. Assume that it does not do so, then

$$\forall l : \lim_{n\to\infty} \gamma^{(n)}_{(i,l)}(t) < p \quad \textit{and} \quad \forall l : \lim_{n\to\infty} \left(\gamma^{(n)}_{(i,l)}(t) + \underbrace{\varepsilon^{(n)}_{t,\lambda}}_{\to 0} \right) > p.$$

But since $\lim_{n\to\infty} \varepsilon^{(n)}_{t,\lambda} = 0$, we obtain

$$p < \gamma_{(i,l)}(t) = \lim_{n\to\infty} \gamma^{(n)}_{(i,l)}(t) < p,$$

which is not possible. Consequently the algorithm will always stop, thereby having computed an N_i for each repeating state, and so the corresponding until formula is level independent as of level $\max_i(N_i)$. For the unbounded until formula, level independence as of level k can be proven similarly. $\square$

3.5 Case study: Connection management

In Section 3.5.1 we describe the *On-demand Connection with Delayed Release* (OCDR) mechanism in detail, in Section 3.5.2 we derive a QBD model for this mechanism. We discuss model checking a steady-state measure in Section 3.5.3 and model checking a measure based on the time-bounded until in Section 3.5.4.

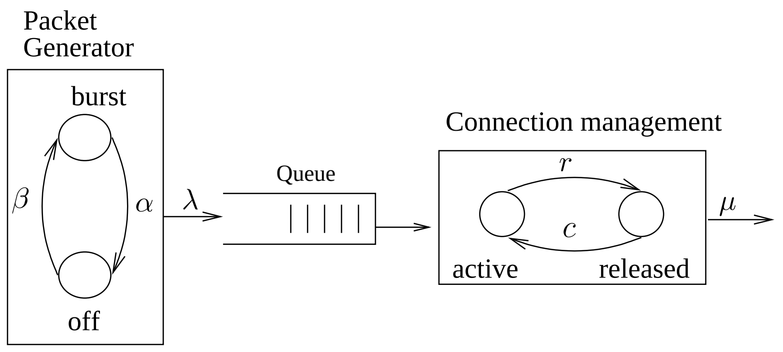


Figure 3.6: Abstract model of the OCDR mechanism

3.5.1 System description

The transport protocol TCP offers a connection-oriented service in the Internet, which implies that a connection should be established prior to any application data is exchanged [52]. For applications with a connectionless nature, such as e-mail or web browsing, prior to application data transfer, a connection needs to be established.

Arriving application-layer protocol data units, e.g., HTTP requests, therefore potentially suffer a delay from connection establishment, unless an existing connection can be (re)used. Once a connection has been established, all application-level packets can be transported and the connection can be released immediately afterwards, or after some delay (time-out). The latter is exactly the mechanism that is being used for HTTP 1.1; its predecessor, HTTP 1.0, does not allow for connection reuse.

We analyze the behavior of the connection management mechanism, known as "on-demand connection with delayed release" (OCDR) [41], as sketched in Figure 3.6. Packets that have to be transported are generated by an abstract *packet generator* and submitted to the transfer queue. The *connection* can be in one of two modes: (i) it can be *active*, so that an arriving packet can be served immediately, at the cost of maintaining a possibly unused connection; (ii) the connection can be *released*, so that an arriving packet can only be transmitted after the connection is re-established, but there are no costs for maintaining an unused connection. Arriving packets at a released connection suffer an extra *connection-establishment delay*. Once active, all queued packets, as well as those newly arriving, will be transported.

In this specific application, the packet generator cycles through periods in which packets are generated with high intensity (in bursts), followed by periods in which no packets are generated at all. The connection management switches between modes, so as to find the right balance between good performance (low delays) and low costs. Having served the last packet of a burst, the connection will be held active for some time. If no new burst starts within some time-out period, the OCDR mechanism

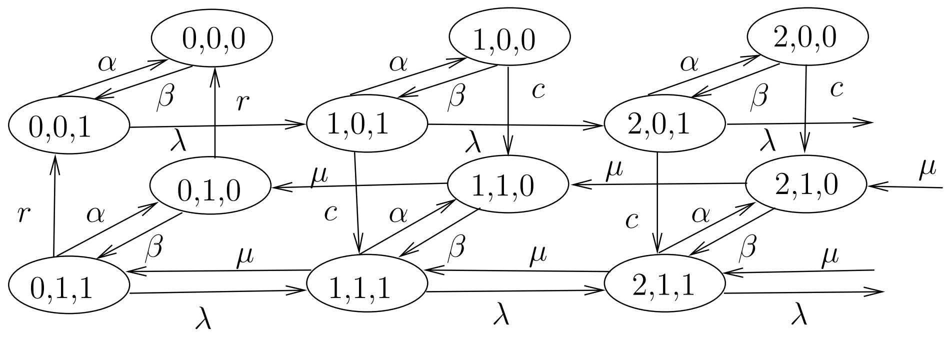


Figure 3.7: QBD model of the OCDR mechanism

releases the connection.

3.5.2 QBD model

To keep the model simple and illustrative, we assume an exponentially distributed connection-establishment delay with rate c, as well as an exponentially distributed time-out for release with mean $1/r$. The model could be extended easily to more deterministic delays, e.g., by using Erlangian approximations [63]. Packets take an exponentially distributed amount of time to be transmitted, with rate μ. In a burst, packets are generated according to a Poisson process with rate λ. The generator switches between epochs of activity and idleness, both exponentially distributed, with rates α and β, respectively. Under these conditions, Figure 3.7 provides the corresponding QBD. In this model the state space is $\mathsf{S} = \{(i,j,k) \mid i \in \mathbb{N}, j,k = 0,1\}$, where i denotes the number of packets queued (and being transmitted), j denotes whether the connection is *active* ($j = 1$) or *released* ($j = 0$), and k denotes whether the packet arrival process is in a burst ($k = 1$) or not ($k = 0$). Clearly, each level i ($= i$ packets present) consists of four states: $S^i = \{(i,0,0),(i,0,1),(i,1,0),(i,1,1)\}$. The symmetric level diameter is $d = 1$. Table 3.1 shows the numerical values of the parameters as presented in [41].

parameter	λ	μ	α	β	c	r
sec^{-1}	100	125	1	0.04	10	10

Table 3.1: Numerical values for the parameters of the model

3.5.3 Model checking steady-state properties

We want to know whether the steady-state probability of being in the different phases with atomic properties $\Phi_1 =$ *active and no burst*, $\Phi_2 =$*released and burst* or

$\Phi_3 =$*active and burst*, is greater than a given probability bound p. For each Φ_i, each level contains exactly one state where this atomic property holds. Hence, the sets $Sat(\Phi_i)$ have infinite size. Figure 3.8 shows the number of iterations (as discussed in Section 3.2.5) needed to verify the property, depending on the probability bound p. If the actual steady-state probability of Φ_i-states comes close to the given bound p, more iterations are needed. This explains the peak at $p = 0.0065$ for Φ_1, at $p = 0.0071$ for Φ_2 and at $p = 0.0313$ for Φ_3. Depending on the chosen probability bound p, the satisfaction sets $Sat(\mathcal{S}_{>p}(\Phi_i))$ are either empty or consist of the complete state space.

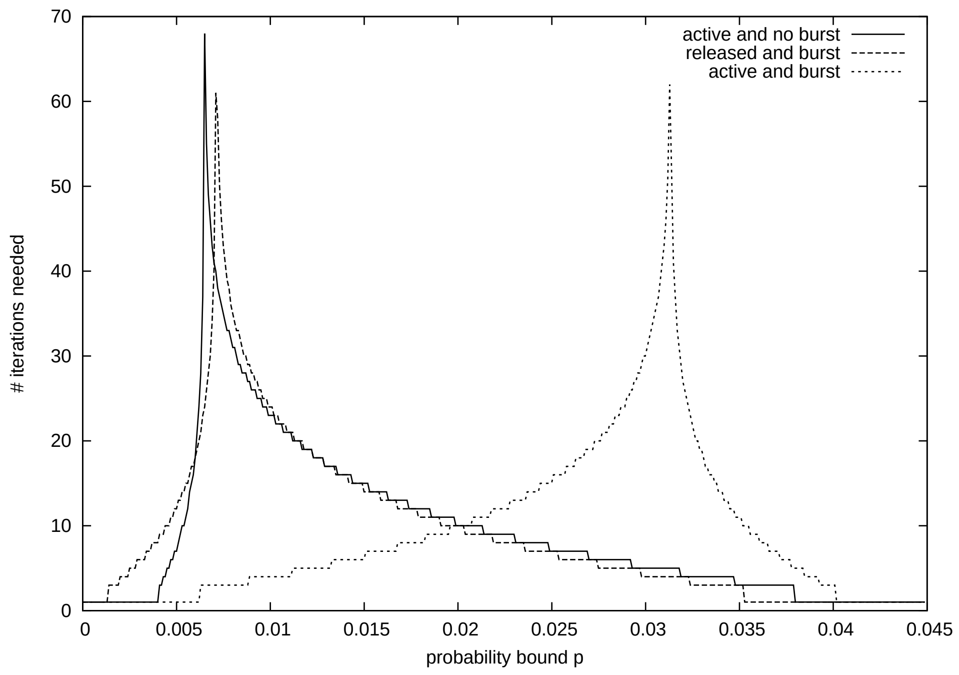


Figure 3.8: Number of iterations needed for checking $\mathsf{s} \models \mathcal{S}_{>p}(\Phi_i), i = 1, 2, 3$

3.5.4 Model checking time-bounded until

Figure 3.9 shows the number of uniformization steps n needed for the computation of $Sat(\mathcal{P}_{\geq p}(tt\ \mathcal{U}^{[0,t]}\Psi))$ for $\Psi =$ *released and no burst* and for $t \in \{0.5, 1.0, 2.0\}$, depending on the probability bound p.

To analyze the efficiency gain using the dynamic termination criterion, as presented in Section 3.4.1, we show the number of iterations with the dynamic termination criterion, as well as the a priori computed number of steps required for an error $\varepsilon_{t,\lambda}^{(n)} = 10^{-4}$. Clearly, the a priori number of steps is independent of the probability bound p and increases with a growing time bound t.

After 0 steps the comparison can be evaluated for $p = 0$ for all time bounds

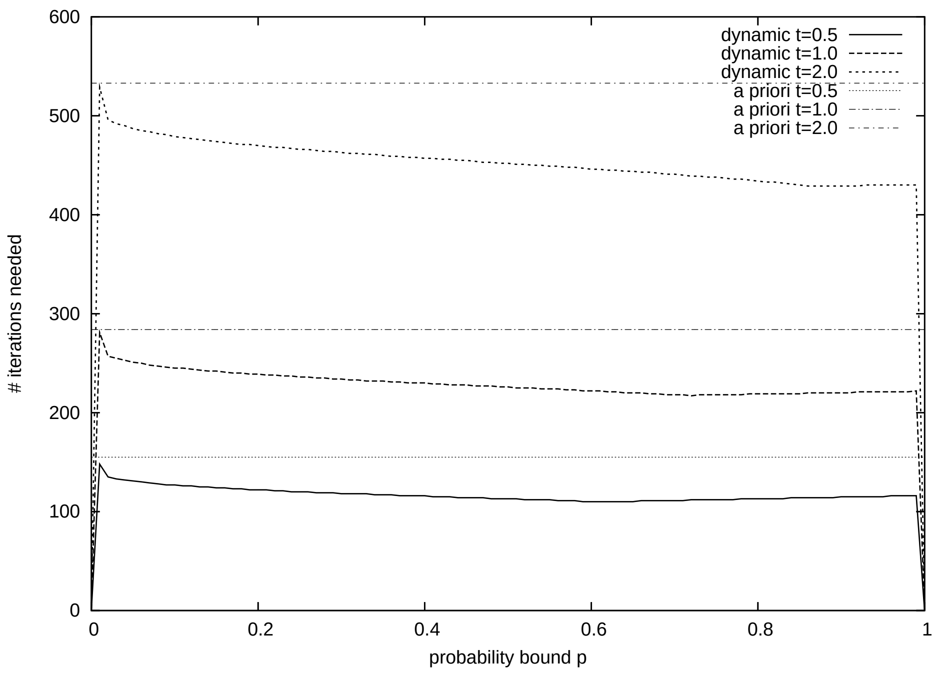


Figure 3.9: Number of iterations needed for checking $\mathsf{s} \models \mathcal{P}_{\geq p}(tt\, \mathcal{U}^{[0,t]} \Psi)$

when using the dynamic termination criterion: every probability is at least 0. For an increasing probability bound p the number of iterations first increases steeply. This is because the Poisson probabilities are de facto equal to zero in the first few iterations and no decision can be made when comparing with any $p > 0$.

Then, for a very low probability bound $p_{\text{peak}} \approx 0.02$, the number of iterations jumps to a peak value for all three time bounds. A peak occurs whenever the computed probability for some state gets really close to the probability bound p we have to compare with. The peak number of iterations with the dynamic termination criterion approximately equals the a priori computed number of steps for $\varepsilon_{t,\lambda}^{(n)} = 10^{-4}$. Hence, we conclude that the computed probability for one of the starting states lies in $[p_{\text{peak}} - 10^{-4}, p_{\text{peak}}]$.

For an increasing probability bound p the number of iteration steps using the dynamic termination criterion decreases. For larger time bounds t the gap between the curves for the dynamic and the a priori number of iteration steps increases, showing the efficiency gain using the dynamic termination criterion.

The execution time per iteration is the same for the dynamic as for the a priori termination criterion. Hence, for the same number of iterations the execution time is the same for one measure. For an increasing number of iterations, the execution time per iteration grows as for large n the iterations take longer, due to the larger matrices involved. For the measure $\mathcal{P}_{\geq p}(tt\, \mathcal{U}^{[0,t]} \Psi)$, the execution time per iteration ranges from $1.26 \cdot 10^{-3}$ sec for the time bound 0.5 to $4.92 \cdot 10^{-3}$ sec for the time bound 2.0. Even though the curves in Figure 3.9 look smooth, small variations in the number of iterations occur when using the dynamic termination criterion.

3.6 Summary

In this chapter we have presented new algorithms for model checking CSL properties against labeled Quasi Birth Death processes. The model checking algorithms make extensive use of uniformization for transient analysis (for time-bounded until) and matrix-geometric methods for determining steady-state probabilities (for the steady-state operator). These model checking algorithms as presented are new.

We are aware of the fact that when checking nested formulas, the number of levels that have level-dependent properties grows, which makes the algorithms less efficient. On the other hand, practice reveals that the nesting depth of logical expressions to be checked is typically small [28], so that this is not so much of a disadvantage after all. By restricting ourselves to level-independent and periodic formulas, we restrict the set of CSL formulas that can be checked. For model checking truly level-*dependent* CSL formulas new model checking algorithms will be needed, since in that case we cannot exploit the level-independent QBD structure to cut the infinite set of states.

Our approach to analyze the transient state probabilities of QBDs with uniformization with representatives is also new. We claim the termination criterion as presented to be optimal, when checking a CSL until-formula. Uniformization with representatives is both computationally and memory efficient. The only drawback the approach suffers from is that the number of considered steps can become large, as n depends on the product λt; but this is a general drawback of uniformization. Large n lead to large matrices $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}$, however, these matrices are very sparse, due to the block-structure. We have shown the feasibility of our approach by a case study.

Chapter 4

CSL model checking algorithms for JQNs

In this chapter we describe CSL model checking algorithms for another class of infinite state Markov chains, namely for the Markov chains that underly labeled *Jackson queueing networks* (JQNs) [67]. We first describe the class of JQNs and the underlying Markov chain in Section 4.1. The general CSL model checking algorithms are then presented in Section 4.2. Section 4.3 then introduces transient analysis on JQNs and the details of model checking the until operator on JQNs are provided in Section 4.4. As a case study, we model an e-business site as JQN and analyze its scalability with the newly developed model checking techniques in Section 4.5. Then, in Section 4.6, we compare the algorithms we have developed for QBDs and JQNs, before we conclude in Section 4.7.

4.1 Model class

Jackson queueing networks are introduced in Section 4.1.1. Section 4.1.2 discusses their underlying infinite state Markov chain. We introduce a simple way to split the underlying infinite state space into an infinite number of finite so-called fronts in Section 4.1.3 and generalize this in Section 4.1.4.

4.1.1 Labeled Jackson queueing networks

A *Jackson queueing network* (JQN) consists of a number of interconnected queueing stations, numbered $1, \ldots, M$. Jobs from an infinite population arrive with a negative exponential inter-arrival time distribution with rate λ at the queueing system. The job service requirements at queue m are negative exponentially distributed, with rate μ_m. There is a single server at each queueing station. This can be generalized easily to m-servers. Jobs arriving at a queue are served in *first come first served*

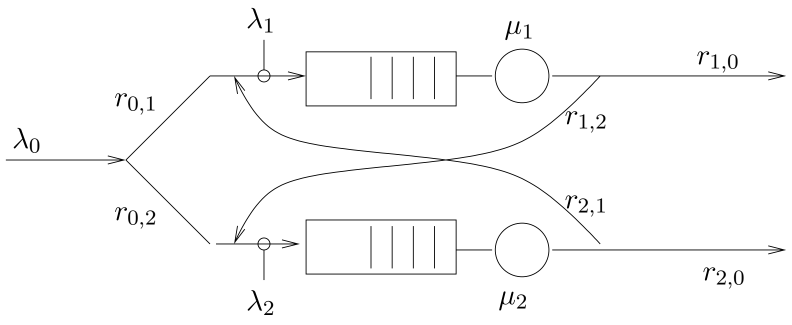


Figure 4.1: JQN with two queues

(FCFS) order. Jobs arriving when the server is busy are queued in an unbounded buffer. Each queueing station in a JQN behaves, in essence, as a simple so-called $M|M|1$ queue. We assume a never empty source from which customers originate and arrive at the JQN, and into which they disappear after having received their service. This environment is indexed 0 and the overall arrival process from the environment is a Poisson process with rate λ.

A finite routing matrix $\mathbf{R} \in [0,1]^{(M+1)\times(M+1)}$ contains the routing probabilities from queue m to queue n with $m, n \in \{0, 1, \cdots, M\}$: $r_{m,n} \in [0,1]$. Note that $\sum_{n=0}^{M} r_{m,n} = 1$, for all m. In case $r_{m,n} = 0$ there is no routing from queue m to queue n, and in case $r_{m,n} = 1$, n is the only output for queue m. The routing probability $r_{m,0}$ gives the probability that a job actually leaves the queueing network after completion at queue m and the routing probability $r_{0,n}$ gives the probability that an arriving customer is routed to queue n. Note that $r_{0,0} = 0$ by definition and that we do allow for direct feedback at the queueing stations (e.g., $r_{m,m} > 0$ is allowed for $m > 0$).

A state in a JQN can be defined as $\mathsf{s} = (s_1, s_2, \cdots, s_M)$, where $s_m \geq 0$ represents the number of customers in queue m. For model checking purposes, we also need a state labeling that distinguishes different classes of states. This leads us to the following definition.

Definition 17 (Labeled Jackson queueing network). A labeled Jackson queueing network **JQN J** of order M (with $M \in \mathbb{N}^+$) is a tuple $(\lambda, \underline{\mu}, \mathbf{R}, L)$ with arrival rate λ, a vector of size M of service rates $\underline{\mu}$, a routing matrix $\mathbf{R} \in \mathbb{R}^{(M+1)\times(M+1)}$ and a labeling function L that assigns a set of valid atomic propositions from a fixed and finite set AP of atomic propositions to each state $\mathsf{s} = (s_1, s_2, \ldots, s_M)$. □

Example 2. In Figure 4.1 we present a JQN with two queues that will serve as running example. The external arrival rate is λ, the vector of service rates is given as $\overline{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ and the routing matrix is $\mathbf{R} = \begin{pmatrix} 0 & r_{0,1} & r_{0,2} \\ r_{1,0} & 0 & r_{1,2} \\ r_{2,0} & r_{2,1} & 0 \end{pmatrix}$. The labeling L will be introduced later.

Definition 18 (Traffic equations and Utilization). The overall rate of jobs arriving at each queue n is given as:

$$\lambda_n = \lambda \cdot r_{0,n} + \sum_{m=1}^{M} \lambda_m \cdot r_{m,n}, \text{ for } n = 1, \ldots, M.$$

These equations are called *(first-order) traffic equations.* The utilization per queue m is defined as $\rho_m = \frac{\lambda_m}{\mu_m}$. □

4.1.2 Underlying Markov chain

The underlying state space of a JQN $\mathbf{J}$ of order M is a highly-structured *labeled infinite state continuous-time Markov chain*, denoted $\mathcal{J}$, with state space $\mathsf{S} = \mathbb{N}^M$, that is infinite in M dimensions. Every state $\mathsf{s} \in \mathsf{S}$ is represented as an M-tuple $\mathsf{s} = (s_1, s_2, \cdots, s_M)$, with $s_m \geq 0$. The labeling function $L : \mathsf{S} \to 2^{AP}$ on the state space then assigns from the set AP of atomic propositions to each state the set of valid atomic propositions. The state $\hat{\mathsf{s}} = (0, \ldots, 0)$ is called *origin.* As the number of customers in queue m is non-negative $s_m \geq 0$, for all $m \in \{1, \ldots, M\}$, the underlying state-space is limited towards the origin in every dimension. The M dimensional state space S is bounded by M so-called *boundary hyperplanes* of dimension $M - 1$. The underlying state space of a JQN is by definition strongly-connected. Note that these boundary hyperplanes consist of an infinite number of states for $M \geq 2$.

State changes may occur due to an arrival at queue m from the environment or a departure to the environment from queue m, or by a service completion at queue m, followed by routing of that customer from queue m to queue n for $1 \leq m, n \leq M$.

- An arrival is always possible, the new state is then defined as $\mathsf{s}' = \mathsf{s} + a$ with $a = (a_1, a_2, \ldots a_M)$. An arrival at queue m, is denoted a with:

$$a_n = \begin{cases} 1, & m = n, \\ 0, & n \neq m, \end{cases} \text{ for } n \in \{1, \ldots, M\}.$$

- A departure or a job routing from queue m is only possible, when there is at least one customer in queue m; in this case the new state s' is computed as: $\mathsf{s}' = \mathsf{s} + d$ or $\mathsf{s}' = \mathsf{s} + f$, respectively. A departure from queue m is denoted as d with:

$$d_n = \begin{cases} -1, & n = m, \\ 0, & n \neq m, \end{cases} \text{ for } n \in \{1, \ldots, M\}.$$

A job routing from queue m to queue n is denoted f with:

$$f_k = \begin{cases} -1, & k = m, \\ 1, & k = n, \\ 0, & \text{otherwise}, \end{cases} \qquad \text{for } k \in \{1, \dots, M\}.$$

By adding a state change vector to the source state, the destination state is defined.

Definition 19 (Transition rate matrix). The rate for a state change from a state s to another state s' within the infinite state space S is given by the highly symmetric transition rate matrix $\mathbf{T}(\mathsf{s}, \mathsf{s}') : \mathsf{S} \times \mathsf{S} \to \mathbb{R}^+$, as follows:

cause	restriction	state change	$\mathbf{T}(\mathsf{s}, \mathsf{s}')$
arrival	none	$\mathsf{s}' = \mathsf{s} + \overline{a}_m$	$\lambda \cdot r_{0,m}$
departure	$s_m > 0$	$\mathsf{s}' = \mathsf{s} + \overline{d}_m$	$\mu_m \cdot r_{m,0}$
routing	$s_m > 0$	$\mathsf{s}' = \mathsf{s} + \overline{f}_{m,n}$	$\mu_m \cdot r_{m,n}$

$\mathbf{T}(\mathsf{s}, \mathsf{s}') = 0$ in all other cases. □

Even though we allow for direct feedback in a JQN, we have to get rid of the self loops in the generator matrix of the underlying CTMC. In [9], Baier et al. show that this transformation preserves the validity of all CSL operators, except for the next operator. This is done by multiplying the service rate μ_m with the probability of leaving queue m, that is $(1 - r_{m,m})$, in case there is at least one job in queue m.

Definition 20 (Generator matrix). The rate for a state change from a state s to another state s', for $\mathsf{s} \neq \mathsf{s}'$ is the same in the generator matrix as in the transition rate matrix:

$$\mathbf{G}(\mathsf{s}, \mathsf{s}') = \mathbf{T}(\mathsf{s}, \mathsf{s}'), \text{ for } \mathsf{s} \neq \mathsf{s}'.$$

The diagonal element $\mathbf{G}(\mathsf{s}, \mathsf{s})$ is defined as the negative sum over all possible outgoing rates from s, that is

$$\mathbf{G}(\mathsf{s}, \mathsf{s}) = -\left(\lambda + \sum_{m=1}^{M} \mu_m \cdot (1 - r_{m,m}) \cdot \mathbf{1}_{(s_m > 0)} \right),$$

where the indicator function $\mathbf{1}_{(s_m > 0)}$ returns 1 if $s_m > 0$, and 0 otherwise. □

Example 3. Figure 4.2 shows the underlying state space of the JQN from Example 2 that is infinite in two dimensions. Arrivals occur in both dimensions with rate $\lambda_1 = \lambda \cdot r_{0,1}$ and $\lambda_2 = \lambda \cdot r_{0,2}$, respectively. Departures with rate μ_1 and μ_2 happen from both dimensions with $\mu_1 \cdot r_{1,0}$ and $\mu_2 \cdot r_{2,0}$, respectively, and jobs are routed from queue 1 to queue 2 with rate $\mu_1 \cdot r_{1,2}$ and from queue 2 to queue 1 with rate $\mu_2 \cdot r_{2,1}$.

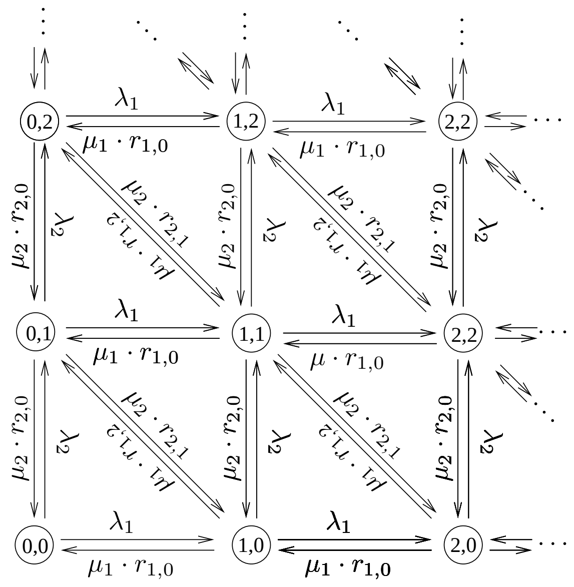


Figure 4.2: Underlying state space of the JQN with two queues from Fig. 4.1

4.1.3 Partitioning the underlying state space

To deal with the underlying infinite state space of JQNs, we need a way to partition the infinite state space into an infinite number of finite sets. Recall that the infinite state space of a QBD is intuitionally partitioned into an infinite number of similar repeating levels. A QBD is infinite in just one dimension, whereas a JQN is infinite in several dimensions, making the partitioning a bit more cumbersome.

In line with the levels of a QBDs we need a partitioning of the state space for JQNs such that with one step only the next higher or the next lower partition of the state space can be reached. Note that infinitely many of such partitionings do exist for JQNs. The easiest partitioning would group all states with the same distance to the origin into the same partition. However, we use a different partitioning that better suits the model checking algorithms as explained in Section 4.2.

The state space of a JQN of order M can be partitioned into infinitely many rectangular shaped fronts, that are pairwise disjoint and situated like shells around each other, as shown in Figure 4.3. The outermost corner $\bar{\imath} = (i, \ldots, i)$ of a front, where all queues have the same number of customers i, is used to uniquely identify a front $F(\bar{\imath})$.

In this two dimensional setting, front $F(\bar{\imath})$ consists of all states that equal i in one dimension and are smaller or equal in the other dimension. $F(\overline{2})$ consists of the following 5 states, where we first list the states that equal i in the first dimension and then the states that equal i in the second dimension:

$$F(\overline{2}) = \{(2,0); (2,1); (2,2); (1,2); (0,2)\}$$

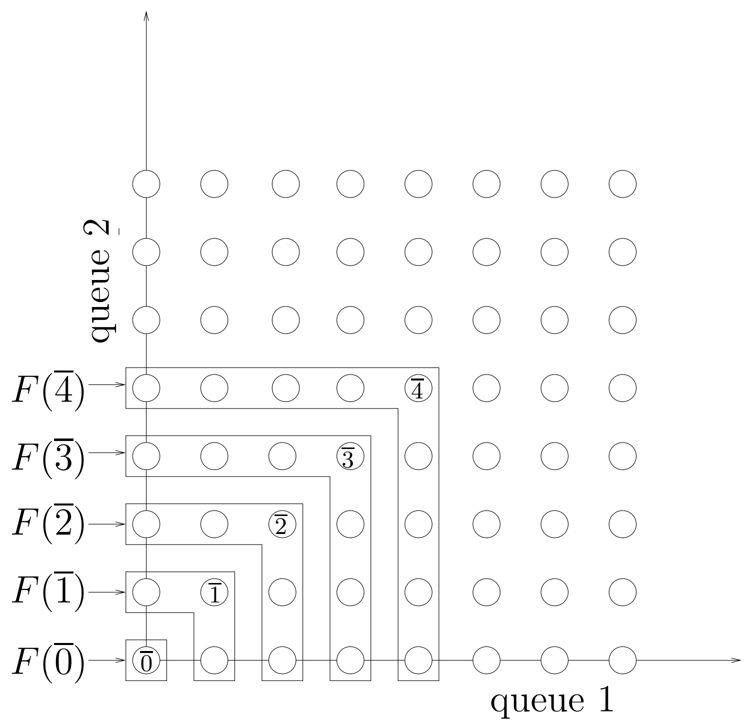


Figure 4.3: Infinite state space of a JQN with 2 queues, partition up in fronts $F(\bar{\imath})$

In an M dimensional JQN, the front $F(\bar{\imath})$ contains all states that equal i in at least one dimension, i.e., $\exists m(s_m = i)$, and are smaller or equal in all remaining dimensions.

Definition 21 (Front $F(\bar{\imath})$**).** In an M dimensional JQN the front $F(\bar{\imath})$ is a finite set of states defined as

$$F(\bar{\imath}) = \{\mathsf{s} \in \mathsf{S} \mid \exists m\ (s_m = i) \wedge (\forall n \neq m\ (s_n \leq i))\}. \qquad \square$$

Lemma 1 (Partitioning of the state space). Every $\mathsf{s} \in \mathsf{S}$ belongs to exactly one front $F(\bar{\imath})$. The infinite state space can then be arranged as

$$\mathsf{S} = \bigcup_{i=0}^{\infty} F(\bar{\imath}) \text{ with } F(\bar{\imath}) \cup F(\bar{\jmath}) = \varnothing \text{ for } i \neq j, i, j \in \mathbb{N}. \qquad \square$$

The number of states per front $F(\bar{\imath})$ increases with i. It can be computed in the M dimensional case, considering that front $F(\bar{\imath})$ consists of those states that are added when extending the M dimensional hypercube with side length i to the M dimensional hypercube with side length $i+1$:

$$|F(\bar{\imath})| = (i+1)^M - i^M. \tag{4.1}$$

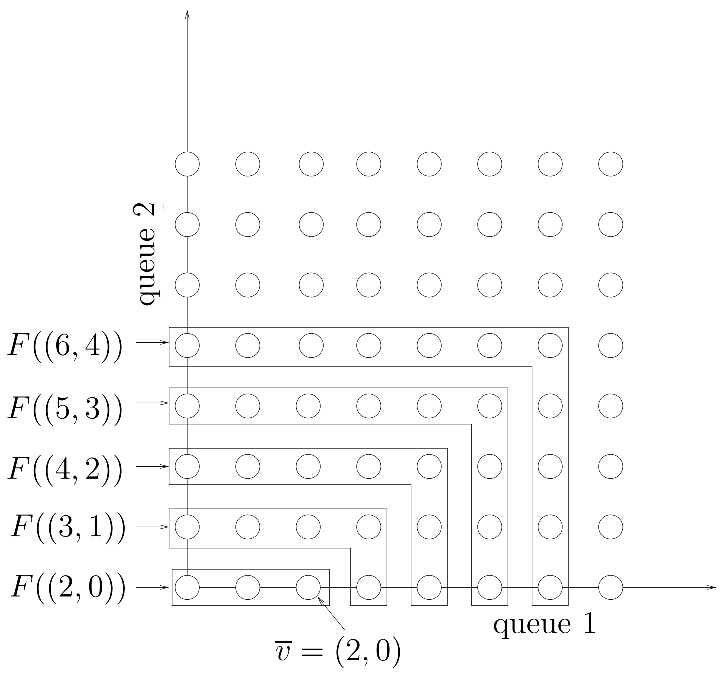


Figure 4.4: General partitioning of the infinite state space of a JQN with two queues, with offset $\overline{v} = (2,0)$

As shown in Figure 4.3, front $F(\overline{4})$ consists of $(4+1)^2 - 4^2 = 9$ states. These are exactly the states that are added to the square with side length 4 to obtain the square with side length 5.

4.1.4 General partitioning

The partitioning as presented in Section 4.1.3 always considers a square part of the infinite state space as the corner points always have the same number of customers in every queue. Basically the underlying state space of an M dimensional JQN can be partitioned in many different rectangular shaped fronts, for an arbitrarily chosen corner point. A given corner point $\overline{v}$ with possibly $v_m \neq v_n$ for $m, n \in \{0, 1, \ldots, M\}$ partitions the infinite state space differently, however, conserving the basic principles of the partitioning. Figure 4.4 shows how the underlying infinite state space of a JQN with two queues that is partitioned with corner point $(2,0)$.

Note that, corner points $\overline{v}$ and $\overline{v} + \overline{\imath}$ for $i \in \mathbb{N}$ obtain the same partitioning. Hence, we need a way to uniquely specify a partitioning.

Definition 22 (Offset). The minimum corner point of a general partitioning is the only corner point that has at least one zero entry. This minimum corner point is denoted offset and uniquely induces a general partitioning. $\square$

As can be seen in Figure 4.4, a partitioning with corner point $(3,1)$ or with $(5,3)$ results in the same partitioning as with offset $\overline{v} = (2,0)$. With a general partitioning, the first front $F(\overline{v})$ may contain more than one state. Front $F(\overline{v}) = F((2,0))$, in Figure 4.4, consists of three states and front $F(\overline{v}+\overline{\imath})$ consists of $3 + i \cdot 2$ states. The state space S can then be partitioned into infinitely many finite sets $F(\overline{v}+\overline{\imath})$, where $i \in \mathbb{N}$:

$$\mathsf{S} = \bigcup_{i=0}^{\infty} F(\overline{v}+\overline{\imath}).$$

Note that given $\overline{v}$ every $\mathsf{s} \in \mathsf{S}$ belongs to exactly one front $F(\overline{v}+\overline{\imath})$.

Definition 23 (General front $F(\overline{v}+\overline{\imath})$). In an M dimensional JQN the front $F(\overline{v}+\overline{\imath})$ is a finite set of states defined as

$$F(\overline{v}+\overline{\imath}) = \{\mathsf{s} \in \mathsf{S} \mid \exists m(s_m = v_m + i) \wedge (\forall n \neq m(s_n \leq v_n + i))\}, \text{with } m \in \{1, \ldots, M\}$$

□

Similar to (4.1), the number of states per front in a general partitioning with minimum corner point $\overline{v}$ equals:

$$|F(\overline{v})| = \prod_{m=1}^{M} (v_m + 1) - \prod_{m=1}^{M} v_m. \tag{4.2}$$

4.2 Model checking algorithms

We now present the general algorithms for CSL model checking labeled JQNs. Independence of atomic propositions is discussed in Section 4.2.1 and independence of CSL formulas is discussed in Section 4.2.2. The general model checking procedure is discussed in Section 4.2.3. In Section 4.2.4 we introduce model checking of logical operators. How to model check the steady-state operator is presented in Section 4.2.6 and how to model check the next operator is shown in Section 4.2.7. Model checking the until operator with its different time bounds is presented in Section 4.2.8.

4.2.1 Independence of atomic propositions

In the following we will restrict ourselves to atomic propositions of the form:

$$ap = \bigwedge_{m=1}^{M} (s_m \bigtriangleup g_m) \text{ for } g_m \in \mathbb{N} \text{ and } \bigtriangleup \in \{<, \geq\}.$$

An atomic proposition ap is thus valid based on the number of jobs per-queue m. The number of jobs per queue has to be either smaller or greater-or-equal than a given per queue threshold g_m. This restricts the formulas we are able to check. Due to this restriction, the validity of an atomic proposition does not change anymore for $s_m \geq g_m$ onwards for dimension m. Hence, we denote $\overline{g} = (g_1, \ldots, g_M)$ *independence vector* and call the atomic proposition *independent as of* $\overline{g}$. For the set of states $\{\mathsf{s} \in \mathsf{S} \mid \forall m(s_m \geq g_m)\}$ the validity of the atomic proposition remains the same.

To induce a partitioning from the independence vector $\overline{g}$, we have to compute the corresponding offset. Thus, we define $g_{\min}$ as the minimum of $g_1, \ldots, g_M$. By subtracting $g_{\min}$ from every entry of $\overline{g}$, the offset is given as: $\overline{v} = \overline{g} - \overline{g_{\min}}$. The offset $\overline{v}$ can then be used to partition the state space as introduced in Section 4.1.4 into fronts $F(\overline{v} + \overline{\mathsf{i}})$ for $i \in \mathbf{N}$. Note that $F(\overline{v} + \overline{\mathsf{i}}) = F(\overline{g})$ for $i = g_{\min}$. The finite set of fronts where the labeling of states may still change is denoted $S_b(\overline{v})$ (for the <u>b</u>oundary set) and is defined as:

$$S_b(\overline{v}) = \bigcup_{i=0}^{g_{\min-1}} F(\widetilde{v} + \overline{\mathsf{i}}).$$

In the following we introduce the concept of representative fronts, states and sets for JQNs, as visualized in Figure 4.5.

Definition 24 (Representative front, state and set). For a JQN and a CSL formula that is independent as of $\overline{g}$ the notion of representatives is defined as follows:

- The front $F(\overline{g})$ is called *representative front* and denoted as $R(\overline{g})$.
- The states in the representative front are called *representative states* $\mathsf{r} \in R(\overline{g})$.
- Each representative state r represents a distinct infinite set of states, denoted S_r. In general, in an M-dimensional JQN, there are M types of representative sets that account for one (1) up to M infinite dimensions. A representative set S_r is called infinite in dimension m if and only if $r_m = g_m$, and restricted in dimension m otherwise. In case a representative state r equals $\overline{g}$ in k dimensions, it represents a k-dimensional set S_r, such that

$$\mathsf{s} \in S_\mathsf{r} \Leftrightarrow \begin{cases} s_m \geq r_m, & \text{iff } r_m = g_m, \\ s_m = r_m, & \text{otherwise.} \end{cases} \qquad \square$$

Hence, a state s belongs to S_r when it takes the same value as r in the restricted dimensions and any value at least r_i in the infinite dimensions. The state space can then also be partitioned into the finite set of boundary states and a finite number of infinite representative sets of states as follows:

$$\mathsf{S} = S_b(\overline{g}) \cup \bigcup_{\mathsf{r} \in R(\overline{g})} S_\mathsf{r}.$$

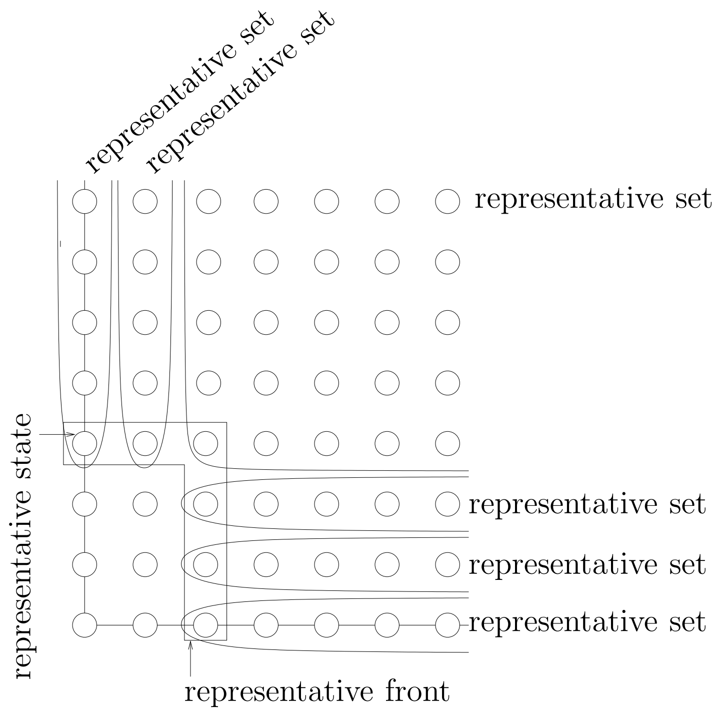


Figure 4.5: Representative front, state and set according to Definition 24.

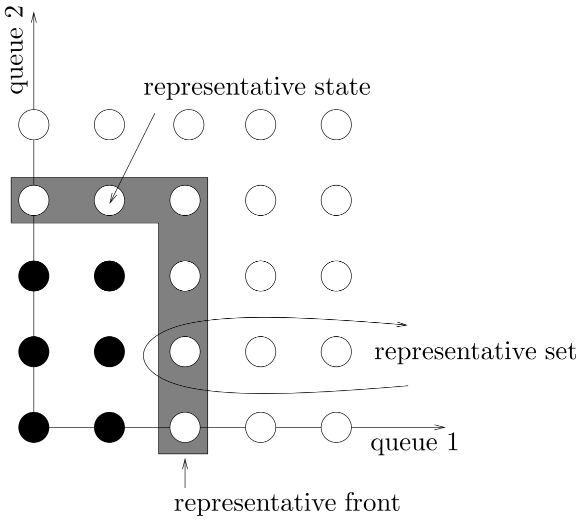
For atomic propositions the representative front can be made smaller, however, for model checking CSL properties in general we need the *full* representative front as defined in Definition 24.

Proposition 8 (Validity of atomic propositions). The validity of an atomic proposition $ap \in AP$ does not change any more in this set, that is,

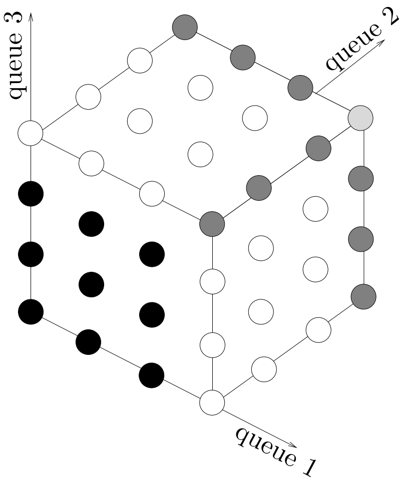
$$L(\mathsf{r}) = L(\mathsf{s}), \text{ for all } \mathsf{s} \in S_{\mathsf{r}}. \qquad \square$$

We will illustrate the use of this concept on atomic propositions, however, it applies to general CSL formula as we will see later.

Example 4. Suppose we define the atomic proposition $\mathrm{ap}_1 = (s_1 \geq 2) \wedge (s_2 \geq 3)$ for the JQN from Example 2. The white states in Figure 4.6(a) depict those states where ap_1 is valid. The atomic proposition is independent as of $\overline{g} = (2, 3)$. The representative front for ap_1 is formed by the states in the grey polygon: $(0, 3)$ accounts for the states $(0, n)$, with $n \geq 3$, and $(1, 3)$ represents the states $(1, n)$ with $n \geq 3$. With $m \geq 2$, $(2, 0)$ represents $(m, 0)$, $(2, 1)$ represents $(m, 1)$ and $(2, 2)$ represents $(m, 2)$, respectively. These five representative states all account for a one dimensional set of states. The representative state $(2, 3)$ accounts for the two-dimensional set of states $S_{(2,3)} = \{\mathsf{s} \in \mathsf{S} \mid s_1 > 2 \wedge s_2 > 3\}$. The black states belong to the boundary set $S_b(2, 3)$ with representative $(0, 0)$.



(a) two-dimensional state space



(b) three-dimensional state space

Figure 4.6: Representative front of states for independent atomic propositions

Example 5. Figure 4.6(b) shows the representative front for the atomic proposition $\text{ap}_2 = (s_1 < 3) \wedge (s_2 < 3) \wedge (s_3 < 3)$ in a JQN of dimension three ($M = 3$). The atomic proposition is valid in the black states only (of the 27 states only 9 are visible). All the remaining depicted states are representative states. We have different types of representative states: the white ones represent a set of states that is infinite in one dimension, the grey ones represent a set that is infinite in two dimensions and the light grey one $(3, 3, 3)$ represents a set that is infinite in three dimensions.

4.2.2 Independence of CSL formulas

Again, we use the logic CSL [8] to express properties for JQNs. The syntax and semantics are the same as for finite CTMCs, with the only difference that we now interpret the formulas over states and paths in JQNs.

From Section 3.2 we know that CSL formulas are not level independent on QBDs in general, even if the atomic propositions are level independent. However, their validity does not change arbitrarily between levels. In the following we will show that for CSL formulas on JQNs we can also find a front from which the validity of the CSL formula does not change anymore.

Definition 25 (Independence of CSL formulas). Let $\mathcal{J}$ be the underlying state space of an JQN of order M. A CSL state formula Φ is *independent* as of $\overline{g}$ if and only if there exists a finite *representative front* $R(\overline{g}) = \{\mathsf{r} \in \mathsf{S} \mid \exists m(r_m = g_m) \wedge (\forall n \neq m(r_n \leq g_n))\}$ such that for all $\mathsf{r} \in R(\overline{g})$ and for all $\mathsf{s} \in S_{\mathsf{r}}$ it holds that $\mathsf{r} \models \Phi \Leftrightarrow \mathsf{s} \models \Phi$. $\square$

Under the assumption of a fixed set of atomic propositions that is independent as of $\overline{g}$, the following proposition states that such a finite representative front $R(\overline{g}')$ exists for any CSL formula. We will justify this proposition inductively over the structure of the logic in Section 4.2.4 for logical operators, in Section 4.2.6 for the steady-state operator, in Section 4.2.7 for the next operator and in Section 4.4.5 for the until operator.

Proposition 9 (Independence on JQNs). Let $\mathcal{J}$ be the underlying state space of a JQN of order M with independent atomic propositions as of $\overline{g}$ and let Φ be a CSL state formula other than $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I \Psi)$. Then there exists a *independence vector* $\overline{g}'$ such that Φ is independent as of $\overline{g}'$ in $\mathcal{J}$.

For the until operator $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I \Psi)$ we assume that for no state s the probability measure is exactly equal to p. Under this assumption, there exists a vector $\overline{g}'$, such that $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I \Psi)$ is independent as of $\overline{g}'$ in $\mathcal{J}$. □

4.2.3 General model checking

For model checking a property Φ, we compute the set $Sat(\Phi)$ with the recursive descent procedure over the parse tree of Φ, as presented in Section 2.6.

Definition 26 (Boundary satisfaction set and representative satisfaction front). For a CSL formula that is independent as of $\overline{g}$, the *boundary satisfaction set* and the *representative satisfaction front* are defined as

$$Sat^{S_b(\overline{g})}(\Phi) = S_b(\overline{g}) \cap Sat(\Phi), \text{ and } Sat^{R(\overline{g})}(\Phi) = R(\overline{g}) \cap Sat(\Phi),$$

respectively. We define the satisfaction set of front i as:

$$Sat^{F(\overline{\imath})}(\Phi) = Sat(\Phi) \cap F(\overline{\imath}).$$ □

The possibly infinite satisfaction set $Sat(\Phi)$ can then be expressed as:

$$Sat(\Phi) = \bigcup_{i=0}^{\infty} Sat^{F(\overline{\imath})}(\Phi).$$

The satisfaction set can also be considered as the union of the *boundary satisfaction set* $Sat^{S_b(\overline{g})}(\Phi)$ and of all representative sets for which the corresponding representative state belongs to the *representative satisfaction front* $Sat^{R(\overline{g})}(\Phi)$:

$$Sat(\Phi) = Sat^{S_b(\overline{g})}(\Phi) \cup \bigcup_{\mathsf{r} \in Sat^{R(\overline{g})}(\Phi)} S_{\mathsf{r}}.$$

Given Φ is independent as of $\overline{g}$, $Sat^{R(\overline{g})}(\Phi)$ acts as a representative for all fronts $F(\overline{g} + \overline{\imath})$ as the validity of Φ does not change any more.

Proposition 10 (Finite representation of the possibly infinite satisfaction set). For a CSL formula Φ that is independent as of $\overline{g}$, we do not need to consider the possibly infinite satisfaction set $Sat(\Phi)$. Instead, it suffices to consider only the level satisfaction sets up to $Sat^{R(\overline{g})}$. $\square$

As explained for atomic propositions, we have to compute the corresponding offset $\overline{v}$ for a given independence vector $\overline{g}$. By subtracting $g_{\min} = \min(g_1, \ldots, g_M)$ from every entry of $\overline{g}$, the offset is given as: $\overline{v} = \overline{g} - \overline{g_{\min}}$. The offset $\overline{v}$ then partitions the state space as introduced in Section 4.1.4. For JQNs, the satisfaction sets, as used in Algorithm 1, can therefore be replaced by the data structure $\bigcup_{i=0}^{g_{\min}} Sat^{F(\overline{v}+\overline{\imath})}$.

4.2.4 Logical operators

The model checking procedure for logical operators is the same as for finite CTMCs. The only thing we need to take care of is how independence changes. Negating a CSL formula does not change its independence. For a CSL formula Φ that is independent as of $\overline{g}$ the negation $\neg\Phi$ is also independent as of $\overline{g}$. However, combining a CSL formula Φ that is independent as of $\overline{g}^{\Phi}$ with a CSL formula Ψ that is independent as of $\overline{g}^{\Psi}$ with conjunction, changes independence depending on the structure of Φ and Ψ. In any case, we can state that $\Phi \wedge \Psi$ is independent as of $\overline{g} = \max\{\overline{g}^{\Phi}, \overline{g}^{\Psi}\}$, where we choose the maximum of g_m^{Φ} and g_m^{Ψ} in every dimension m. Note that this new independence vector $\overline{g}$ might be larger than necessary, depending on the structure of Φ and Ψ.

4.2.5 Steady-state distribution for JQNs

Recall that utilization per queue m is defined as $\rho_m = \lambda_m/\mu_m$. We require stability of JQNs as follows: In case all $\rho_m = \lambda_m/\mu_m < 1$, the QN is said to be stable. Then, the number of arriving jobs per unit time is smaller than the amount of jobs that each queue can handle per unit time. This guarantees that the queue will not build up infinitely large. In the following we restrict ourselves to stable JQNs, to be able to compute steady-state probabilities. As we require stable queues the underlying state space of the JQN is strongly connected and, the initial state does not influence the steady-state probabilities.

The long-run probability that s customers are presently in a single $M|M|1$ queue, that is for a single stable $M|M|1$ queue (with $\rho = \frac{\lambda}{\mu} < 1$) is: $\Pr\{S = \mathsf{s}\} = (1-\rho)\rho^s$, where S is the random variable indicating the number of customers in the queue [51]. In [46, 47], Jackson proved the following theorem.

Theorem 1 (Jackson [46]). The overall steady-state probability distribution in the underlying state space of a stable JQN $\mathcal{J}$ is the product of the per-queue steady-state probability distributions, where the queues can be regarded as if operating independently from each other:

$$\Pr\{S = s\} = \prod_{m=1}^{M} (1 - \rho_m)(\rho_m)^{s_m}, \qquad \mathsf{s} = (s_1, \ldots, s_M) \in \mathsf{S}. \qquad \square$$

4.2.6 Steady-state operator

A state s satisfies $\mathcal{S}_{\bowtie p}(\Phi)$ if the sum of the steady-state probabilities of all Φ-states reachable from s meets the bound p. Since a stable JQN is by definition strongly connected, the steady-state probabilities are independent of the starting state. It follows that either all states satisfy a steady-state formula or none of the states does, which implies that a steady-state formula is always independent as of $\overline{g} = \hat{\mathsf{s}} = (0, \ldots, 0)$.

We sum the steady-state probabilities of all states that satisfy Φ by summing over all states in the representative sets $\mathsf{s}' \in S_\mathsf{r}$ for all representatives $\mathsf{r} \models \Phi$ that satisfy Φ and over all states in the boundary set $S_b(\overline{g})$ that satisfy Φ:

$$\mathsf{s} \models \mathcal{S}_{\bowtie p}(\Phi) \Leftrightarrow \left(\sum_{\mathsf{s}' \in Sat^{S_b(\overline{g})}(\Phi)} \pi(\mathsf{s}, \mathsf{s}') + \sum_{\mathsf{r} \in Sat^{R(\overline{g})}(\Phi)} \sum_{\mathsf{s}' \in S_\mathsf{r}} \pi(\mathsf{s}, \mathsf{s}') \right) \bowtie p \tag{4.3}$$

We obtain $Sat(\mathcal{S}_{\bowtie p}(\Phi)) = \mathsf{S}$, if the accumulated steady-state probability meets the bound p, otherwise $Sat(\mathcal{S}_{\bowtie p}(\Phi)) = \varnothing$. In case the representative state r is in $Sat(\Phi)$, all states $\mathsf{s}' \in S_\mathsf{r}$ are in $Sat(\Phi)$. The accumulated steady-state probability for all states $\mathsf{s}' \in S_\mathsf{r}$ is given as a product over the finite and infinite dimensions of S_r:

$$\sum_{\mathsf{s}' \in S_\mathsf{r}} \pi(\mathsf{s}, \mathsf{s}') = \prod_{m=1}^{M} \Omega(m), \text{ with } \Omega(m) = \begin{cases} (1 - \rho_m)(\rho_m)^{r_m}, & \text{for } r_m \neq g_m, \\ (\rho_m)^{g_m}, & \text{for } r_m = g_m. \end{cases} \tag{4.4}$$

In this expression we distinguish between the finite and the infinite dimensions of a representative set S_r. In a finite dimension m ($r_m \neq g_m$), we multiply with $(1 - \rho_m)(\rho_m)^{r_m}$ and in an infinite dimension m ($r_m = g_m$)we multiply with $(\rho_m)^{g_m}$.

Proof (Equation 4.4). Recall that

$$\mathsf{s}' \in S_\mathsf{r} \Leftrightarrow \begin{cases} s'_m \geq r_m & \text{iff } r_m = g_m, \\ s'_m = r_m & \text{otherwise.} \end{cases}$$

Applying Jackson's theorem, the accumulated steady-state probability is given by

$$\sum_{\mathsf{s}' \in S_\mathsf{r}} \pi(\mathsf{s}, \mathsf{s}') = \sum_{\mathsf{s}' \in S_\mathsf{r}} \left(\prod_{m=1}^{M} (1 - \rho_m)(\rho_m)^{s'_m} \right). \tag{4.5}$$

According to Jackson's theorem we can consider the dimensions independently from each other. Hence, for every dimension $m = 1, \ldots, M$, a state $\mathsf{s}' \in S_{\mathsf{r}}$ may take all values $n \geq g_m$ in case $r_m = g_m$ and the value r_m in case $r_m \neq g_m$. Thus,

$$\sum_{\mathsf{s}' \in S_{\mathsf{r}}} \pi(\mathsf{s}, \mathsf{s}') = \prod_{m=1}^{M} \Omega(m), \text{ with } \Omega(m) = \begin{cases} \sum_{n=g_m}^{\infty} (1-\rho_m)(\rho_m)^n, & \text{for } r_m = g_m, \\ (1-\rho_m)(\rho_m)^{r_m}, & \text{for } r_m \neq g_m. \end{cases} \tag{4.6}$$

The infinite sum can be rewritten

$$\sum_{n=g_m}^{\infty} (1-\rho_m)(\rho_m)^n = (1-\rho_m) \sum_{n=g_m}^{\infty} (\rho_m)^n = (1-\rho_m) \frac{(\rho_m)^{g_m}}{1-\rho_m} = (\rho_m)^{g_m}$$

and replaced to match (4.4):

$$\Omega(m) = \begin{cases} (1-\rho_m)(\rho_m)^{r_m}, & \text{for } r_m \neq g_m, \\ (\rho_m)^{g_m}, & \text{for } r_m = h_m. \end{cases}$$

□

The model checking routine for the steady-state operator is stated in pseudo code in Algorithm 10. First the steady-state probabilities for the Φ-states in the boundary set are accumulated according to Theorem 1, then the steady-state probability for the remaining infinite state-space is computed according to Equation (4.4).

Example 6. We want to check the CSL formula $\mathcal{S}_{\bowtie p}((s_1 \geq 2) \wedge (s_2 \geq 3))$. Recall from Example 4 that all states $\mathsf{r} \in R((2,3)) = \{(0,3),(1,3),(2,0),(2,1),(2,2),(2,3)\}$ satisfy $ap_1 = (s_1 \geq 2) \wedge (s_2 \geq 3)$ and the boundary states do not satisfy ap_1. Using (4.3) and (4.4) and accumulating the probabilities for all representative states $\mathsf{r} \in R((2,3))$ we obtain

$$\begin{aligned} &\mathsf{s} \models \mathcal{S}_{\bowtie p}((s_1 \geq 2) \wedge (s_2 \geq 3)) \Leftrightarrow \\ &\big((1-\rho_1)\rho_2^3 + (1-\rho_1)\rho_1\rho_2^3 + \rho_1^2(1-\rho_2) + \rho_1^2(1-\rho_2)\rho_2 + \rho_1^2(1-\rho_2)\rho_2^2 + \rho_1^2\rho_2^3\big) \bowtie p. \end{aligned} \tag{4.7}$$

Given concrete values for λ_i and μ_i, the utilizations $\rho_i = \lambda_i/\mu_i$ can be computed and we can easily check this inequality.

Algorithm 10 $Sat_{\mathcal{S}}(\bowtie p, \Phi) : Sat^{F(\widetilde{0})}$

```
begin
$\Phi$ independent as of $\overline{g}$;
$\pi_b = 0$;
for all $s \in Sat^{S_b}(\Phi)$ do
  $\pi_b \mathrel{+}= \prod_{m=1}^{M}(1-\rho_m)(\rho_m)^{s_m}$;                (* according to Theo. 1 *)
end for
$\pi = 0$;
for all $r \in Sat^{R(\overline{g})}$ do
  $\pi_r = 1$;
  for all $i \in \{1, \ldots, M\}$ do
    $\pi_r = \pi_r \cdot \Omega(m)$;                                         (* according to Eq. (4.4) *)
  end for
  $\pi \mathrel{+}= \pi_r$;
end for
if $\pi_b + \pi \bowtie p$ then
  return $F(\overline{0})$;
else
  return $\varnothing$;
end if
end
```

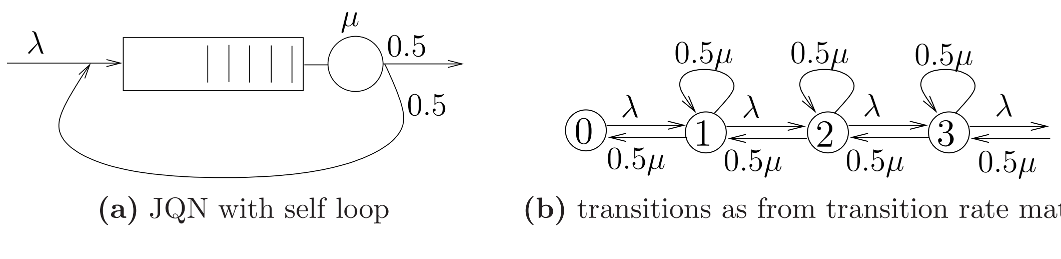
4.2.7 Time-bounded next operator

The time-bounded next operator for JQNs is computed just as for QBDs [68]. Recall that a state s satisfies $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ if the one-step probability to reach a state that fulfills Φ within a time $t \in [t_1, t_2]$, outgoing from s meets the bound p. As for one-step probabilities self loops have to be taken into account, we have to use the transition rate matrix $\mathbf{T}$ to model check the time-bounded next operator. The possibly infinite summation over all Φ-states can be truncated by only considering those Φ-states that can be reached in one step from s. We define the set of states that is reachable in one step from state s as $B_\mathsf{s} = \{\mathsf{s}' \in \mathsf{S} \mid \mathbf{T}(\mathsf{s},\mathsf{s}') > 0\}$. Note that B_s is always finite. We then have:

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi) &\Leftrightarrow \Pr\{\sigma \in Path(\mathsf{s}) \mid \sigma \models \mathcal{X}^{[t_1,t_2]}\Phi\} \bowtie p \\ &\Leftrightarrow \left(\left(e^{-\mathbf{E}(\mathsf{s})\cdot t_1} - e^{-\mathbf{E}(\mathsf{s})\cdot t_2}\right) \cdot \sum_{\mathsf{s}' \in Sat(\Phi)\cap B_\mathsf{s}} \frac{\mathbf{T}(\mathsf{s},\mathsf{s}')}{\mathbf{E}(\mathsf{s})} \right) \bowtie p, \end{aligned} \tag{4.8}$$

where $e^{-\mathbf{E}(\mathsf{s})\cdot t_1} - e^{-\mathbf{E}(\mathsf{s})\cdot t_2}$ is the probability of leaving s at a time $t \in [t_1, t_2]$, and $\mathbf{T}(\mathsf{s},\mathsf{s}')/\mathbf{E}(\mathsf{s})$ specifies the probability to step from state s to state s', provided a step takes place.

Now, let Φ be independent as of $\overline{g}$. Hence, the validity of Φ might be different for all states $\mathsf{s} \in S_b(\overline{g})$. Therefore, the representative states $\mathsf{r} \in R(\overline{g})$ may satisfy $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$, whereas the remaining states $\mathsf{s} \in S_\mathsf{r}$ do not necessarily satisfy $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$, since $S_b(\overline{g})$ is reachable in one step. However, from $\overline{g} + \overline{1}$ onwards, only states with equivalent Φ validity can be reached by a single step. Thus, in case Φ is independent as of $\overline{g}$, $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ is independent as of $\overline{g} + \overline{1}$. For the construction of the satisfaction set of such a formula, we have to compute explicitly the satisfying states in $S_b(\overline{g})$. $Sat^{R(\overline{g}+\overline{1})}(\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi))$ then provides the validity of $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[t_1,t_2]}\Phi)$ for the remaining infinite state space $\mathsf{S} \setminus S_b(\overline{g})$.



(a) JQN with self loop **(b)** transitions as from transition rate matrix

Figure 4.7: Example JQN to illustrate why self loops have to be taken into account when checking the next operator

Example 7. Figure 4.7(a) shows a simple JQN with one queue and direct feedback. Figure 4.7(b) illustrates the underlying state space with transitions according to the transition rate matrix $\mathbf{T}$. We want to check whether the formula $\mathcal{P}_{\bowtie p}(\mathcal{X}^{[0,t]} \geq 2)$

is valid in state 2. Only states 2 and 3 fulfill ≥ 2 and have a non-zero transition probability starting from state 2. Note that $\mathbf{E}(2) = \mu + \lambda$, $\mathbf{T}(2,2) = 0.5\mu$ and $\mathbf{T}(2,3) = \lambda$. According to Equation 4.8 we have to check whether:

$$\left(1 - e^{-(\lambda+\mu)\cdot t}\right) \frac{0.5\mu + \lambda}{\mu + \lambda} \bowtie p.$$

Using $\mathbf{G}$ instead of $\mathbf{T}$ in (4.8) would lead to checking whether:

$$\left(1 - e^{-(\lambda+0.5\mu)\cdot t}\right) \frac{\lambda}{0.5\mu + \lambda} \bowtie p,$$

as self loops are not considered. Depending on the value of p, this may lead to a different validity of $2 \models \mathcal{P}_{\bowtie p}(\mathcal{X}^{[0,t]} \geq 2)$.

4.2.8 Time-bounded until operator

For model checking $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^I \Psi)$ we adopt the general approach for finite CTMCs [8] and QBDs, as presented in Section 3.2.7 and [72]. We have chosen to recapitulate the approaches for model checking the until operator for the different time bounds, to keep this chapter self-contained. Recall that the CSL path formula $\varphi = \Phi\, \mathcal{U}^I \Psi$ is valid if a Ψ-state is reached on a path during the time interval I via only Φ-states. We discuss model checking the until operator for the time intervals $[0,t]$, $[t_1,t_2]$, $[t,t]$, $[0,\infty)$ and $[t,\infty)$. These four cases are basically the same as presented for QBDs in Section 3.2.7. However, the infinite state space of JQNs is partition differently, which is reflected in the following.

The justification for Proposition 9 for the until operators is postponed to Section 4.4.5, as we need a better understanding of how the probabilities are calculated first.

Case $I = [0,t]$

For time intervals of the form $I = [0,t]$, the future behavior of the JQN is irrelevant for the validity of φ, as soon as a Ψ-state is reached. Thus all Ψ-states can be made absorbing without affecting the satisfaction set of formula φ. On the other hand, as soon as a $(\neg\Phi \wedge \neg\Psi)$-state is reached, φ will be invalid, regardless of the future evolution.

As a result of the above consideration, we may switch from checking the underlying CTMC $\mathcal{J}$ to checking a new, derived, Markov chain denoted as $\mathcal{J}[\Psi][\neg\Phi\wedge\neg\Psi] = \mathcal{J}[\neg\Phi \vee \Psi]$, where all states in the underlying Markov chain that satisfy the formula in square brackets are made absorbing. The generator matrix $\tilde{\mathbf{G}}(\mathsf{s},\mathsf{s}')$ for $\mathcal{J}[\neg\Phi \vee \Psi]$ is then defined as

$$\tilde{\mathbf{G}}(\mathsf{s},\mathsf{s}') = \begin{cases} \mathbf{G}(\mathsf{s},\mathsf{s}'), & \mathsf{s} \not\models \neg\Phi \vee \Psi, \\ 0, & \text{otherwise}, \end{cases} \tag{4.9}$$

for $\mathsf{s} \neq \mathsf{s}'$. $\tilde{\mathbf{G}}(\mathsf{s},\mathsf{s})$ is adapted such that it contains the negative sum over all outgoing rates from s.

Proposition 11 (Connectivity of absorbing $\mathcal{J}$). Given a JQN $\mathcal{J}$ and a CSL formula Φ that is independent as of $\overline{g}$, the Markov chain $\mathcal{J}[\Phi]$ is not necessarily a JQN, as it might not be strongly connected, anymore. □

Model checking a formula involving the until operator then reduces to calculating the transient probabilities $\pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s},\mathsf{s}',t)$ for all Ψ-states s'. Exploiting the partitioning of the underlying state space as presented in Section 4.1.3 then yields:

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\,\mathcal{U}^{[0,t]}\Psi) &\Leftrightarrow \mathit{Prob}^{\mathcal{J}}(\mathsf{s},\Phi\,\mathcal{U}^{[0,t]}\Psi) \ \bowtie\ p \\ &\Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s},\mathsf{s}',t) \right) \bowtie p. \end{aligned} \tag{4.10}$$

The transient probabilities are accumulated for the Ψ states in fronts $F(\bar{\imath})$ for $i \in \mathbf{N}$. The transient probability of being in each state of the infinite-state JQN for any possible initial state can be calculated with a new iterative uniformization-based method, which we will present in the Section 4.3. To calculate the satisfaction set for $\mathcal{P}_{\bowtie p}(\Phi\,\mathcal{U}^{[0,t]}\Psi)$, we need to understand how this algorithm works, therefore we postpone this discussion to Section 4.4.1 .

Case $I = [t_1, t_2]$

Considering a time interval $[t_1, t_2]$ with $0 < t_1 < t_2$ we can split the computation in two parts. The first part then addresses the path from the starting state s via Φ-states to a Φ-state s' at time t_1. The second part of the computation addresses the path from s' to a Ψ-state s'' via only Φ states. This leads us to two transformed Markov chains: $\mathcal{J}[\neg\Phi]$ that is used in the first part and $\mathcal{J}[\neg\Phi \vee \Psi]$ in the second part. To calculate the probability for such a path, we accumulate the transition probabilities for all triples $(\mathsf{s},\mathsf{s}',\mathsf{s}'')$, where $\mathsf{s}' \models \Phi$ is reached before time t_1 and $\mathsf{s}'' \models \Psi$ is reached before time $t_2 - t_1$. This can be done, because the underlying Markov chains are time homogeneous.

$$\begin{aligned} &\mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\,\mathcal{U}^{[t_1,t_2]}\Psi) \Leftrightarrow \mathit{Prob}^{\mathcal{J}}(\mathsf{s},\Phi\,\mathcal{U}^{[t_1,t_2]}\Psi) \ \bowtie\ p \\ &\Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s},\mathsf{s}',t_1) \cdot \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}',\mathsf{s}'',t_2-t_1) \right) \bowtie p. \end{aligned} \tag{4.11}$$

The algorithm for the bounded until operator with interval $I = [t_1, t_2]$ will be presented in Section 4.4.2.

Case $I = [t, t]$

The point interval until is a simplification of the interval until, where only the first part of the computation needs to be taken into account. Thus, we need the transformed QBD $\mathcal{J}[\neg\Phi]$ and need to compute the probability that at time point t a state s' is reached that fulfills $\Phi \wedge \Psi$.

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[t,t]} \Psi) \Leftrightarrow & \mathit{Prob}^{\mathcal{J}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,t]} \Psi) \ \bowtie \ p \\ & \Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in \mathit{Sat}^{F(\bar{\imath})}(\Phi \wedge \Psi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \right) \bowtie p. \end{aligned} \tag{4.12}$$

The algorithm for the point interval until operator is the same as for the time bounded until, with two minor changes. First the transient probabilities have to be computed on $\mathcal{J}[\neg\Phi]$ for the point interval until and on $\mathcal{J}[\neg\Phi \vee \Psi]$ for the time bounded until. Second, the goal states s' have to fulfill $\Phi \wedge \Psi$ for the point interval until and just Ψ for the time bounded until.

Case $I = [0, \infty)$

For the unbounded case, the probability $\mathit{Prob}^J(\mathsf{s}, \Phi\mathcal{U}^{[0,\infty)}\Psi)$ equals the probability to eventually reach a Ψ-state. Since the $\neg\Phi \vee \Psi$-states are absorbing, this is exactly the steady-state probability to be in a Ψ-state in the adapted underlying Markov chain, so we have

$$\begin{aligned} & \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,\infty)} \Psi) \Leftrightarrow \mathit{Prob}^J(\mathsf{s}, \Phi\, \mathcal{U}^{[0,\infty)} \Psi) \bowtie p \Leftrightarrow \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(s, \mathit{Sat}(\Psi)) \bowtie p \\ & \Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in \mathit{Sat}^{F(\bar{\imath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}, \mathsf{s}') \right) \bowtie p. \end{aligned} \tag{4.13}$$

The algorithm for the unbounded until operator with time interval $I = [0, \infty)$ will be discussed in Section 4.4.3.

Case $I = [t, \infty)$

For the interval $[t, \infty)$ the computation is split in two parts, just as for $[t_1, t_2]$. The first part addresses the path from the starting state s to a Φ-state s' via only Φ-states at time t, whereas the second part addresses the path that eventually leads from s' to a Ψ-state. Note that we combine the transient probabilities in the transformed JQN $\mathcal{J}[\neg\Phi]$ for the first part with the steady-state probabilities in $\mathcal{J}[\neg\Phi \vee \Psi]$ for the second part as follows:

$$\begin{aligned} \mathsf{s} \models \mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[t,\infty)}\Psi) &\Leftrightarrow \mathit{Prob}^{\mathcal{J}}(\mathsf{s}, \Phi\, \mathcal{U}^{[t,\infty)}\Psi) \ \bowtie\ p \\ \Leftrightarrow \left(\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(n \cdot \bar{\jmath})}(\Psi)} \right. & \left. \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \pi^{\mathcal{J}[\neg\Phi \vee \Psi]}(\mathsf{s}', \mathsf{s}'') \right) \bowtie p. \end{aligned} \tag{4.14}$$

The algorithm for the unbounded until operator with time interval $I = [t, \infty)$ will be introduced in Section 4.4.4.

4.3 Uniformization with Representatives

As we have seen uniformization with representatives is an efficient method to compute the transient probabilities on QBDs. In this section we will adapt this method to be able to compute the transient probabilities on JQNs.

In Section 4.3.1 we describe how to apply the principle of uniformization to JQNs. Section 4.3.2 then describes how to obtain a finite data representation for JQNs and addresses the growth of the involved data structures. The actual iterative algorithm is then presented in Section 4.3.3 before we discuss complexity issues in Section 4.3.4.

4.3.1 Uniformization on JQNs

Recall that the main principles of uniformization for finite and infinite CTMCs have been described already in Section 3.3.1. We have shown there how transient probabilities can be computed for QBDs using uniformization by considering only a finite fraction of the infinite state space. As standard property of uniformization, the finite time bound t is transformed to a finite number of steps n [36].

To do the same for JQNs, first the probability matrix $\mathbf{P}(\mathsf{s}, \mathsf{s}')$ for the uniformized DTMC is defined as

$$\mathbf{P}(\mathsf{s}, \mathsf{s}') = \frac{\mathbf{G}(\mathsf{s}, \mathsf{s}')}{\nu} \text{ for } \mathsf{s} \neq \mathsf{s}', \text{ and } \mathbf{P}(\mathsf{s}, \mathsf{s}) = \frac{\mathbf{G}(\mathsf{s}, \mathsf{s})}{\nu} + 1, \text{ for all } \mathsf{s}, \mathsf{s}'.$$

The uniformization constant ν must be at least equal to the maximum of absolute values of $\mathbf{G}(\mathsf{s}, \mathsf{s})$; for JQNs, the value $\nu = \lambda + \sum_{m=1}^{M} \mu_m$ suffices. For an allowed maximum numerical error $\varepsilon_{t,\nu}^{(n)}$, uniformization requires a finite number n of steps (state changes) to be taken into account in order to compute the transient probabilities; n can be computed *a priori*, given $\varepsilon_{t,\nu}^{(n)}$, ν and t, as for finite CTMCs and QBDs.

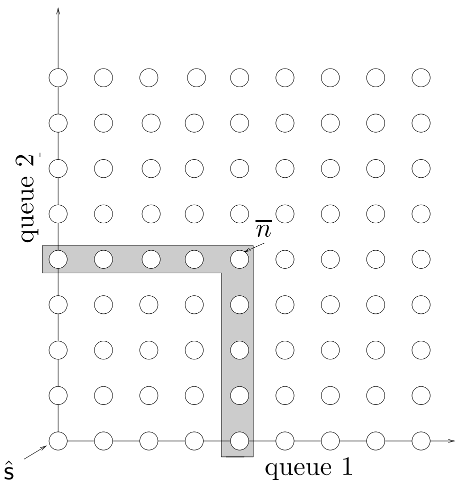


Figure 4.8: Finite state space that needs to be considered for a given $\overline{l} = (4, 4)$

4.3.2 Finite representation

In the following we will use uniformization to compute the transient probabilities to reach all possible goal states from all (starting) states in a JQN. The homogeneous probability matrix $\mathbf{P}$ contains the probability to reach a state s' from a state s within one step for all $\mathsf{s}, \mathsf{s}' \in \mathsf{S}$. From every possible starting state, only n fronts can be reached with n steps. Hence, for a given number of steps n, all states that are $n+1$ steps away from the origin $\hat{\mathsf{s}}$ are identical in the JQN. We only need to consider a finite part of the JQN, depending on the number of steps n: For a given number of steps n it is sufficient to consider all states in the first $n+1$ fronts as starting states and the first $2n+1$ fronts as goal states. $R(\overline{n+1})$ is then the representative front.

As we will see, the homogenous structure of the JQN and of the probability matrix implies that we obtain identical transient probabilities for states $\mathsf{s}, \mathsf{s}' \in S_{\mathsf{r}}$ with $\mathsf{r} \in R(\overline{l})$, within the error bounds of uniformization given n steps. In fact, we restrict the computation to a finite number of starting states and still perform a comprehensive transient analysis for every possible state as starting state.

Example 8. As shown in Figure 4.8, starting from every representative state $\mathsf{r} \in R(\overline{n})$, still l steps can be undertaken in every direction without reaching beyond the origin $\hat{s}$. The total amount of starting states we have to consider equals $(n+1)^2$ and the total amount of goal states equals $(2n+1)^2$. According to Equation (4.1), the representative front $R(\overline{n})$ contains $(n+1)^2 - n^2 = 2n+1$ states. Thus, $2n+1$ states of the starting states represent the remaining infinite state space.

Proposition 12 (Starting states and goal states). In an M dimensional setting, for a given number of n steps, $(n+1)^M$ starting states and $(2n+1)^M$ goal states have to be considered out of which $(n+1)^M - l^M$ states are representative. $\square$

As for the transient analysis of QBDs (Section 3.3.1), the matrix $\mathbf{U}^{(n)}$ is the state probability matrix after n discrete epochs and $\mathbf{V}^{(n)}(t)$ holds the approximated transient probabilities after n steps. Note that these matrices remain two-dimensional for JQNs, as they represent all possible combinations of starting states and goal states.

Similar to Section 3.3.2, it is now sufficient to consider only starting states that belong to fronts $F(\bar{\imath})$ for $i \leq n$ for a finite representation of $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$. The size of the finite representation depends on the considered number of steps n, hence, on the time, the uniformization rate, and the required accuracy.

We now address the growth of the matrices $\mathbf{U}^{(n)}$ in the course of the computation. Figure 4.9(a) shows that the dimension of the finite representation of $\mathbf{U}^{(0)}$ is:

$$\dim(\mathbf{U}^{(0)}) = (|F(\overline{0})|)^2 = (1^M - 0^M)^2 = 1.$$

Since $n = 0$, we cannot leave a state and the first front $R(\overline{0})$ is already a representative front.

Figure 4.9(b) shows the dimension of the finite representation of $\mathbf{U}^{(1)}$. Since $n = 1$, we can reach the next higher or the next lower fronts. Thus, front $F(\overline{0})$ cannot be used as representative front, but we can use the next higher front $R(\overline{1})$ as representative front, as shown in Figure 4.9(b). Since $n = 1$, it is possible to reach the front $F(\overline{2})$ as well; thus we have to consider starting in one of the first two fronts $F(\bar{i})$ for $i = 0, 1$ and ending up in one of the first three fronts $F(\bar{j})$ for $j = 0, 1, 2$. The dimension of the finite representation of $\mathbf{U}^{(1)}$ depends on the fronts that contain the starting states and on the fronts containing the goal states. The number of states of a given front can be calculated according to Equation (4.2). The dimension of $\mathbf{U}^{(1)}$ is given by:

$$\begin{aligned}\dim(\mathbf{U}^{(1)}) &= \big(|F(\overline{0})| + |R(\overline{1})|\big) \times \big(|F(\overline{0})| + |R(\overline{1})| + |F(\overline{2})|\big)\\ &= \big(1^M - 0^M + 2^M - 1^M\big) \times \big(1^M - 0^M + 2^M - 1^M + 3^M - 2^M\big)\\ &= 2^M \times 3^M.\end{aligned}$$

Figure 4.9(c) shows the finite representation of the matrix $\mathbf{U}^{(2)}$. From a given front, we can reach at most two more fronts in both directions. Picking the second front as new representative, ensures that we cannot reach beyond the origin $\hat{s}$. We have to attach another row of states to represent starting from the new representative front. Furthermore, we attach two more columns to account for the fronts $F(\overline{3})$ and $F(\overline{4})$ that can now be reached from the new representative front. The dimension of the finite representation then equals

$$\begin{aligned}\dim(\mathbf{U}^{(2)}) &= \big(|F(\overline{0})| + |F(\overline{1})| + |R(\overline{2})|\big) \times \big(|F(\overline{0})| + |F(\overline{1})| + |R(\overline{2})| + |F(\overline{3})| + |F(\overline{4})|\big)\\ &= 3^M \times 5^M.\end{aligned}$$

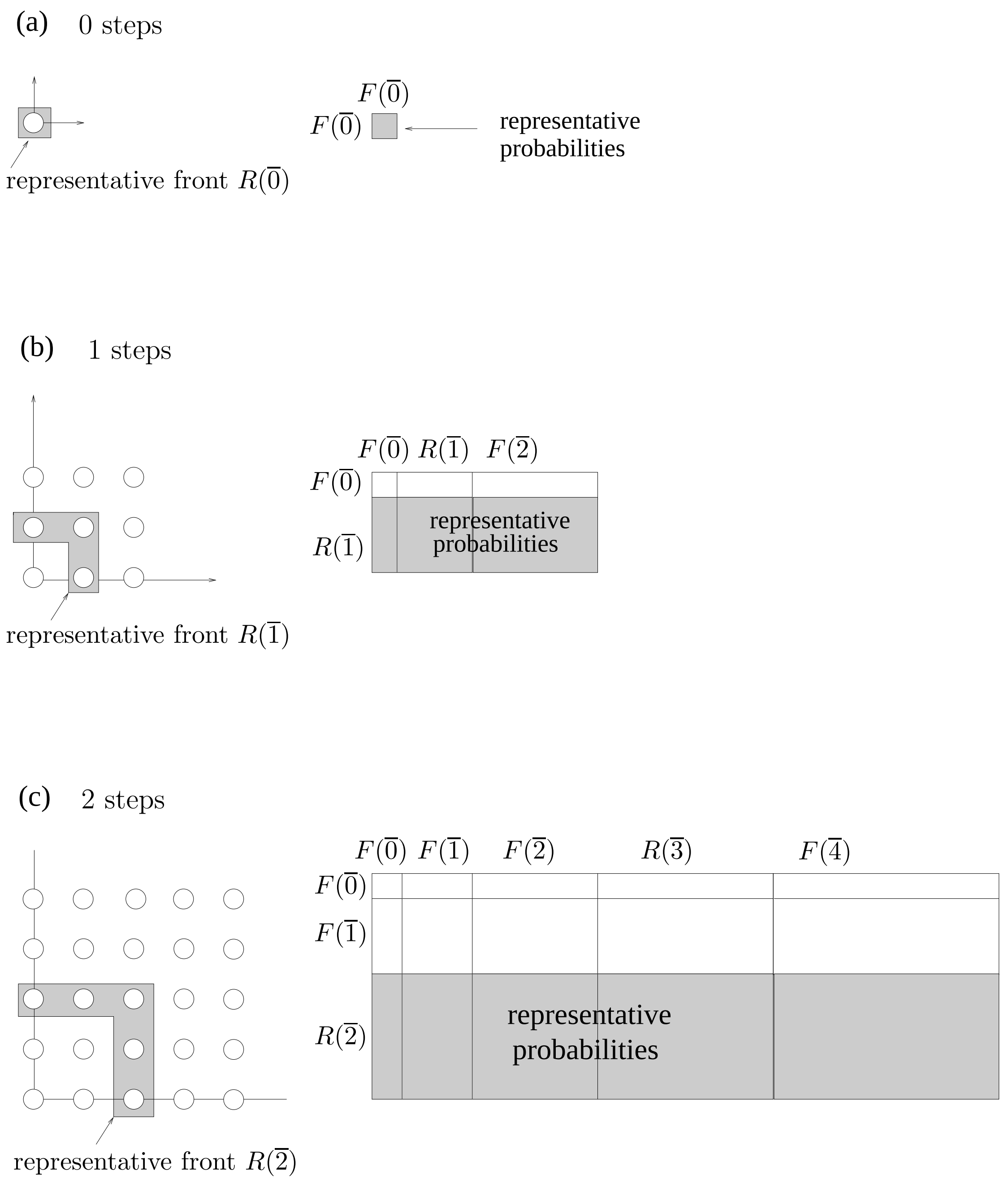

(a) 0 steps
$F(\overline{0})$
$F(\overline{0})$
representative probabilities
representative front $R(\overline{0})$
(b) 1 steps
$F(\overline{0})$ $R(\overline{1})$ $F(\overline{2})$
$F(\overline{0})$
$R(\overline{1})$
representative probabilities
representative front $R(\overline{1})$
(c) 2 steps
$F(\overline{0})$ $F(\overline{1})$ $F(\overline{2})$ $R(\overline{3})$ $F(\overline{4})$
$F(\overline{0})$
$F(\overline{1})$
$R(\overline{2})$
representative probabilities
representative front $R(\overline{2})$

Figure 4.9: Considered part of the state space (left) and finite representation of $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$ (right), depending on the number of considered steps

In a general JQN, for a given number of steps n, the size of the matrix $\mathbf{U}^{(n)}$ is then

$$\dim(\mathbf{U}^{(n)}) = \left(\sum_{i=0}^{n} |F(\bar{i})|\right) \times \left(\sum_{j=0}^{2\cdot n} |F(\bar{j})|\right) = (n+1)^M \times (2\cdot n+1)^M. \qquad (4.15)$$

Note that, even though the left side of Figure 4.9 only shows the two dimensional case, $(M = 2)$ the right side is also correctly depicted for an M dimensional setting. As before, the finite representation of the matrix $\mathbf{V}^{(n)}(t)$ has the same dimension as $\mathbf{U}^{(n)}$.

4.3.3 Uniformization with Representatives

We now proceed with the actual computation of the state probability matrix $\mathbf{U}^{(n)}$ and the approximated transient probability matrix $\mathbf{V}^{(n)}(t)$ according to (3.10). Starting with $n = 0$, and thus with the smallest finite portion of the JQN, cf. Figure 4.9(a), we increase n step by step, thus increasing accuracy and size of the considered finite representation of the JQN. However, in each iteration we always use the smallest possible representation.

Considering n steps, the probability of starting in a state in the representative front $\mathsf{r} \in R(\overline{n})$ and ending in a state s' in one of the fronts $F(\bar{i})$, for $i \in \{0, \ldots, 2\cdot n\}$, represents the probability of starting in a state $\mathsf{s} \in S_{\mathsf{r}}$ and ending in the corresponding state s''.

In order to increase the number of steps from $n-1$ to n we first adapt the size of the data structure before computing the values for n steps. Moving from step $n-1$ to n we have to add the front $F(\overline{n})$ that is going to be representative for n steps, to the set of starting states and the fronts $F(2\overline{n} - \overline{1})$ and $F(2\overline{n})$ to the set of goal states.

First, the two new sets of columns of goal states are initialized with zero, as it is impossible to reach these states with $n-1$ steps. Second, the new row of starting states $F(\overline{n})$ is initialized with the probabilities of the corresponding entries from front $R(\overline{n} - \overline{1})$ that is representative for $n-1$ steps. Note that this holds for $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$.

An entry $(\mathsf{s}, \mathsf{s}')$ in the new row of starting states $F(\overline{n})$ constitutes moving from a starting state s to a goal state s' with $\mathsf{s} \in F(\overline{n})$ and $\mathsf{s}' \in F(\bar{\imath})$ for $i = 0, \ldots 2n$. We first need to find the corresponding starting state $\mathsf{r} \in R(\overline{n} - \overline{1})$ such that $\mathsf{s} \in S_{\mathsf{r}}$. The corresponding goal state then is the state s'' that is, in every dimension, exactly as far away from r than s' is from s, that is, $\mathsf{r} - \mathsf{s}'' = \mathsf{s} - \mathsf{s}'$.

Proposition 13 (Corresponding tuples). Given a tuple of starting and goal state $(\mathsf{s}, \mathsf{s}')$ with $\mathsf{s} \in F(\overline{n})$, the corresponding tuple $(\mathsf{r}, \mathsf{s}'')$ with $\mathsf{r} \in R(\overline{n-1})$ is given by:

$$\mathsf{r} = \mathsf{s} - \overline{h}(\mathsf{s}) \text{ and } \mathsf{s}'' = \mathsf{s}' - \mathsf{s} + \mathsf{r},$$

with

$$\overline{h}(\mathsf{s}) = \begin{cases} h_i = 1, & s_i = n, \\ h_i = 0, & s_i \neq n. \end{cases}$$

□

The matrices $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}(t)$ have a block structure, according to the fronts of a JQN; we denote the blocks that give the probabilities from states in front $F(\bar{\imath})$ to states in front $F(\bar{\jmath})$ as $\mathbf{U}^{(n)}_{\bar{\imath},\bar{\jmath}}$ and $\mathbf{V}_{\bar{\imath},\bar{\jmath}}(t)$. Note that $\mathbf{P}$ can also be organized according to this block structure. In iteration step n, we then need to multiply the enlarged representation of $\mathbf{U}^{(n-1)}$ with the square part of $\mathbf{P}$ that accounts for the one-step probabilities for all states in the first $2 \cdot n$ fronts. In general, for $n \geq 1, \mathbf{U}^{(n)}$ is computed as $\mathbf{U}^{(n-1)} \cdot \mathbf{P}$, cf. (3.7), as follows:

$$\mathbf{U}^{(n)}_{\bar{\imath},\bar{\jmath}} = \sum_{k=0}^{2n+1} \mathbf{U}^{(n-1)}_{\bar{\imath},\overline{k}} \cdot \mathbf{P}_{\overline{k},\bar{\jmath}}, \tag{4.16}$$

for $\bar{\imath} = \overline{0}, \ldots, \overline{n}$ and $\bar{\jmath} = \overline{0}, \ldots, 2 \cdot \overline{n}$.

Proposition 14 (Corresponding blocks). For a given number of steps n and for all $i > n$ and for all $j \geq n + i$, the entries of blocks $\mathbf{U}^{(n)}_{\bar{i},\bar{j}}$ equal the representative probabilities from block $\mathbf{U}^{(n)}_{\overline{n},\overline{\jmath-\imath+n}}$, for each corresponding tuple, according to Proposition 13. Hence, our finite representation is correct. □

Proof. The proof of the above proposition follows the same inductive lines as the proof of Proposition 7. □

Due to the block structure of $\mathbf{V}^{(n)}(t)$, we can rewrite (3.9) as:

$$\mathbf{V}^{(n)}(t) = \mathbf{V}^{(n-1)}(t) + \psi(\lambda t; n) \cdot \mathbf{U}^{(n)}, \tag{4.17}$$

again for $\bar{\imath} = \overline{0}, \ldots, \overline{n}$ and $\bar{\jmath} = \overline{0}, \ldots, 2 \cdot \overline{n} + \overline{1}$.

4.3.4 Complexity issues

In the k-th iteration, we actually consider the states of the first k fronts as starting states and the states of the first $2 \cdot k$ fronts as goal states, resulting in matrices with $(k+1)^M \times (2 \cdot k + 1)^M$ entries, as given by (4.15).

If n is the maximum number of steps considered, the overall storage complexity for the three probability matrices $\mathbf{U}^{(n-1)}$, $\mathbf{U}^{(n)}$, $\mathbf{V}^{(n)}$ and the discrete transition matrix $\mathbf{P}$ is $\mathcal{O}(4n^{2 \cdot M})$.

The k-th multiplication of matrix $\mathbf{U}^{(n-1)}$ with $\mathbf{P}$ is carried out in $\mathcal{O}(k^{6M})$. For n the maximum number of considered steps, the overall time complexity therefore equals $\mathcal{O}(n^{6 \cdot M+1})$.

Note that the iteration costs per step increase. However, when full probability matrices of the size $\mathbf{U}^{(n)}$ and $\mathbf{V}^{(n)}$ are used throughout the complete computation, the iteration costs are much higher.

Both the storage complexity and the computational complexity for doing uniformization with representatives on JQNs is much higher than for uniformization with representatives on QBDs. This is due to the fact that the state space of a QBD grows without bound in just one direction, whereas the state space of a JQNs grows without bound in as many directions as the JQN has queues. In a QBD we add for every step the same amount of states, whereas we add in a JQN with every step a growing amount of states, that depends on M. Therefore, it is not possible to give a block-tri diagonal representation for the matrix $\mathbf{P}$ and in step n we have to consider three matrices of dimension $(n+2)^M \times (2 \cdot n + 2)^M$ and one matrix of dimension $(2 \cdot n + 2)^{2 \cdot M}$.

4.4 Different time intervals for the until operator

The algorithms for model checking the until operator for the different time intervals are almost the same on JQNs than on QBDs. The only difference is that the infinite state space is partitioned in fronts instead of levels. Hence, the only difference with the algorithms presented for QBDs is that Sat^i changes to $Sat^{F(\bar{\imath})}$ and $\mathcal{Q}$ changes to $\mathcal{J}$. Note that, as for QBDs, the algorithms, presented in the following, do not stop in case the computed probability for at least one starting state equals the probability bound p.

The algorithm to compute the satisfaction set of a CSL formula with the bounded until operator for the time interval $I = [0, t]$ will be introduced in Section 4.4.1. In Section 4.4.2 we present the algorithm for the interval until $I = [t_1, t_2]$. The algorithms for the unbounded until operator with time interval $I = [0, \infty)$ and time interval $I = [t, \infty)$ are discussed in Section 4.4.3 and in Section 4.4.4, respectively. Furthermore, we provide a proof of Proposition 9 for the until operator in Section 4.4.5.

4.4.1 The simplest case $I = [0, t]$

As the complexity for uniformization with representatives on JQNs is worse than for QBDs it is even more important to use the dynamic termination criterion, as presented in Section 3.4.1 for QBDs.

For model checking an until-formula $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{[t_1,t_2]}\Psi)$ we have to compare for each starting state the probability to follow a $(\Phi\ \mathcal{U}^{[t_1,t_2]}\Psi)$-path with the probability bound p. In the transformed JQN $\mathcal{J}[\neg\Phi \vee \Psi]$ the set of goal states consists of all Ψ-states. We denote the probability to end up in a Ψ-state, given starting state s, as $\gamma_{\mathsf{s}}(t)$. For the time interval $I = [0, t]$, we have:

$$\gamma_{\mathsf{s}}(t) = \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\overline{g}+\overline{\imath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi \vee \Psi]}(\mathsf{s}, \mathsf{s}', t).$$

Note that the vector $\gamma(t)$ consists of sub-vectors corresponding to the fronts of the JQN. The approximation of $\gamma_{\mathsf{s}}(t)$ after n iterations is denoted $\gamma^{(n)}(t) = \mathbf{V}^{(n)}(t) \cdot \gamma(0)$, with

$$\gamma_s(0) = \begin{cases} 1, & \mathsf{s} \models \Psi, \\ 0, & \text{otherwise.} \end{cases}$$

In principle, $\gamma^{(n)}(t)$ is of infinite size, but we can cut it to a finite representation, as from a representative front on, corresponding states have the same probability values. For all states $\mathsf{s} \in \mathsf{S}$, we add the computed transient probabilities to reach any Ψ-state and check whether the accumulated probability meets the bound p on a regular basis.

The accumulated probability is always an underestimation of the actual probability. Recall that $\varepsilon_{t,\nu}^{(n)}$ is the maximum error of uniformization after n iteration steps (cf. (3.11)), such that $\gamma_{\mathsf{s}}(t) \leq \gamma_{\mathsf{s}}^{(n)}(t) + \varepsilon_{t,\nu}^{(n)}$ for time interval $I = [0, t]$. From (3.11) it follows that the value of $\varepsilon_{t,\nu}^{(n)}$ decreases as n increases. Exploiting the above inequality, we obtain the following termination criteria:

$$\begin{array}{lrcl} \text{(a)} & \gamma_{\mathsf{s}}^{(n)}(t) \geq p & \Rightarrow & \gamma_{\mathsf{s}}(t) \geq p, \\ \text{(b)} & \gamma_{\mathsf{s}}^{(n)}(t) < p - \varepsilon_{t,\nu}^{(n)} & \Rightarrow & \gamma_{\mathsf{s}}(t) < p. \end{array}$$

These criteria can be exploited as follows. Starting with a small number of steps, we check whether for the current approximation one of the inequalities (a) or (b) holds for all starting states. If this is not the case we continue, check again, etc., until either of the termination criteria holds. However, if for one of the starting states $\mathsf{s} \in \mathsf{S}$ we have $\gamma_{\mathsf{s}}(t) = p$, the iteration never stops, as neither of the termination criteria ever holds. However, this is highly unlikely to occur in practice.

In case $(\neg\Phi \vee \Psi)$ is independent as of $\overline{g}$ and either (a) or (b) holds for all considered starting states with n steps, front $R(\overline{g} + \overline{n})$ is representative and the transient probabilities for all $\mathsf{s} \in S_{\mathsf{r}}$ computed with n steps will be the same. $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]} \Psi)$ then is independent as of $\overline{g}+\overline{n}$. In that case, we check for all states $\mathsf{s} \leq \overline{g}+\overline{n}$ whether the accumulated transient probability of reaching a Ψ-state meets the bound p. The representative states that satisfy $\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]} \Psi)$ form the representative satisfaction set $Sat^{R(\overline{g}+\overline{l})}(\mathcal{P}_{\bowtie p}(\Phi\, \mathcal{U}^{[0,t]} \Psi))$.

Recall, that the point interval until can be computed just like the time bounded until, with the only difference that the transient probabilities have to be computed on $\mathcal{J}[\neg\Phi]$ and that the goal states s' have to fulfill $\Phi \wedge \Psi$.

4.4.2 The bounded case $I = [t_1, t_2]$

As presented in Section 3.2.7, the computation of the transient probabilities for the time interval $I = [t_1, t_2]$, is split into two parts. We need to combine the transient probabilities in the transformed $\mathcal{J}[\neg\Phi]$ at time t_1 with the transient probabilities in $\mathcal{J}[\neg\Phi \vee \Psi]$ at time $t_2 - t_1$, as follows:

$$
\begin{aligned}
\gamma_{\mathsf{s}}(t_2) &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1) \cdot \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1) \\
&= \underbrace{\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t_1)}_{} \cdot \underbrace{\sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'', t_2 - t_1)}_{} \\
&= \qquad\qquad \gamma_{\mathsf{s}}(t_1) \qquad\qquad \cdot \qquad\qquad \gamma_{\mathsf{s}'}(t_2 - t_1)
\end{aligned}
$$

The first equation is rewritten by separating the transient probabilities of the two parts. For a given a priori error ε_{t_1,ν_1} an initial distribution $\gamma_{\mathsf{s}}^{n_1}(t_1)$ for the first part of the transient probabilities is computed with uniformization in n_1 steps. The second part of the transient probabilities is computed with dynamic uniformization with representatives. The approximation of $\gamma_{\mathsf{s}'}(t_2 - t_1)$ with n_2 steps is denoted as $\gamma_{\mathsf{s}'}^{n_2}(t_2 - t_1)$ and induces the error $\varepsilon_{t_2-t_1,\nu_2}^{(n_2)}$. Both approximations are an underestimation of the actual probability:

$$
\gamma_{\mathsf{s}}(t_2) \leq \left(\gamma_{\mathsf{s}}^{(n_1)}(t_1) + \varepsilon_{t_1,\nu_1}^{(n_1)}\right) \cdot \left(\gamma_{\mathsf{s}'}^{(n_2)}(t_2 - t_1) + \varepsilon_{t_2-t_1,\lambda_2}^{(n_2)}\right). \tag{4.18}
$$

For a given initial distribution $\gamma_{\mathsf{s}}^{n_1}(t_1)$, we iterate n_2 until for every considered starting state one of the following termination criteria holds:

$$
\begin{array}{lrcl}
\text{(a)} & \gamma_{\mathsf{s}}^{n_1}(t_1) \cdot \gamma_{\mathsf{s}'}^{n_2}(t_2 - t_1) > p & \Rightarrow & \gamma_{\mathsf{s}}(t_2) > p \\
\text{(b)} & \gamma_{\mathsf{s}'}^{n_2}(t_2 - t_1) < \frac{p}{\gamma_{\mathsf{s}}^{n_1} + \varepsilon_{t_1,\nu_1}^{(n_1)}} - \varepsilon_{t_2-t_1,\nu_2}^{(n_2)} & \Rightarrow & \gamma_{\mathsf{s}}(t_2) < p.
\end{array}
$$

Given $\neg\Phi \vee \Psi$ is independent as of $\bar{l}$, we have to find n_2 such that either (a) or (b) is fulfilled for every considered starting state. Given that the initial distribution is computed with an a priori known number of steps n_1, $R(\bar{l} + \overline{n}_1 + \overline{n}_2)$ serves as representative for all fronts larger than $\bar{l} + \overline{n}_1 + \overline{n}_2$.

4.4.3 The unbounded case $I = [0, \infty)$

In Section 4.2.8 we showed how the model checking of an unbounded until operator relies on the computation of the steady-state probabilities in the absorbing JQN

$\mathcal{J}[\neg\Phi \vee \Psi]$. Note that $\mathcal{J}[\neg\Phi \vee \Psi]$ is not necessarily a JQN anymore. So we cannot compute the steady-state probabilities using Jackson's theorem. However, the steady-state probability of the set of absorbing Ψ-states is independent of the residence time in each state-visit but only depends on the branching probabilities and the starting state.

We can therefore switch to the discrete-time Markov chain that is derived from the embedded discrete-time Markov that corresponds to $\mathcal{J}[\neg\Phi \vee \Psi]$, by removing self-loops. This probability matrix is denoted $\mathbf{H}$ where

$$\mathbf{H}(\mathsf{s},\mathsf{s}') = \mathbf{G}(\mathsf{s},\mathsf{s}')/-\mathbf{G}(\mathsf{s},\mathsf{s}) \text{ for } \mathsf{s} \neq \mathsf{s}' \text{ and } \mathbf{H}(\mathsf{s}',\mathsf{s}') = 0.$$

Note that $\mathbf{H}^n(\mathsf{s},\mathsf{s}')$ denotes the probability to reach s' from s in exactly step n. If the starting state is from front $F(\bar{l})$, the desired steady-state probability is:

$$\begin{aligned}\pi(\mathsf{s}, Sat(\Psi)) &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Psi)} \pi(\mathsf{s},\mathsf{s}') \\ &= \sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Psi)} \sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s},\mathsf{s}') \\ &= \sum_{n=0}^{\infty} \sum_{i=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}').\end{aligned}$$

The second equality replaces the steady-state probability $\pi(\mathsf{s},\mathsf{s}')$ by the probability to reach a Ψ state s' in zero up to infinitely many steps $\sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s},\mathsf{s}')$.

The third equality follows from the fact that only a finite number of fronts is reachable in n steps. Using this equation, representative probabilities can be computed for an increasing number of considered steps n as is the case for uniformization. The termination criterion is adopted from Section 3.2.5. In every step we have to compare the approximations computed with N steps:

$$\tilde{\pi}^{(N)}(\mathsf{s}, Sat(\Psi)) = \sum_{n=0}^{N} \sum_{i=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Psi)} \mathbf{H}^n(\mathsf{s},\mathsf{s}')$$

and $\tilde{\pi}^N(\mathsf{s}, Sat(\neg\Psi))$, with p and $1-p$, respectively, until we can stop the iteration. As soon as one of the left-hand side inequalities becomes true, we can stop:

$$\begin{array}{llll}\text{(a)} & \tilde{\pi}^{(N)}(Sat(\Phi)) > p & \Rightarrow & \pi(Sat(\Phi)) > p, \\ \text{(b)} & \tilde{\pi}^{(N)}(Sat(\neg\Phi)) > 1-p & \Rightarrow & \pi(Sat(\Phi)) < p.\end{array}$$

Given Ψ is independent as of $\bar{l}$, we have to find N, such that either (a) or (b) holds for all considered starting states. Front $R(\bar{l}+\overline{N})$ is then representative for all fronts larger $\bar{l}+\overline{N}$.

4.4.4 The unbounded case $I = [t, \infty)$

.

Model checking the time-bounded until operator for the time interval $I = [t, \infty)$, consists of two parts. We need to combine the transient probabilities in the transformed JQN $\mathcal{J}[\neg\Phi]$ at time t with the steady-state probabilities in $\mathcal{J}[\neg\Phi \vee \Psi]$. If the starting state is from front $F(\bar{l}$, the desired steady-state probability is:

$$
\begin{aligned}
&\pi(s, Sat(\Phi))\\
=&\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'')\\
=&\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \pi^{\mathcal{J}[\neg\Phi\vee\Psi]}(\mathsf{s}', \mathsf{s}'')\\
=&\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \sum_{j=0}^{\infty} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \sum_{n=0}^{\infty} \mathbf{H}^n(\mathsf{s}', \mathsf{s}'')\\
=&\sum_{i=0}^{\infty} \sum_{\mathsf{s}' \in Sat^{F(\bar{\imath})}(\Phi)} \pi^{\mathcal{J}[\neg\Phi]}(\mathsf{s}, \mathsf{s}', t) \cdot \sum_{n=0}^{\infty} \sum_{j=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \mathbf{H}^n(\mathsf{s}, \mathsf{s}').
\end{aligned}
$$

The first equation can be rewritten by separating the transient from the steady-state probabilities. The steady-state probabilities are expressed via the embedded discrete-time Markov chain with probability matrix $\mathbf{H}$.

To actually compute $\pi(s, Sat(\Phi))$, an initial distribution for the steady-state probabilities, denoted $\widetilde{\pi}_s(t)$ is computed for a given error $\varepsilon_{t,\nu}^{(n)}$. The steady-state probabilities are approximated as follows:

$$
\widetilde{\pi}^{(N)}(s', Sat(\Phi)) = \sum_{n=0}^{N} \sum_{j=\max(0,l-n)}^{l+n} \sum_{\mathsf{s}'' \in Sat^{F(\bar{\jmath})}(\Psi)} \mathbf{H}^n(\mathsf{s}, \mathsf{s}').
$$

Combining the initial distribution with the approximated steady-state probabilities $\widetilde{\pi}^N(s', Sat(\Phi))$ for increasing N, we compare with p and combining the initial distribution with $\widetilde{\pi}^N(s', Sat(\neg\Phi))$ we compare with $1-p$ for every N, until we can stop the iteration. As soon as one of the left-hand side inequalities becomes true, we can stop:

$$
\text{(a)} \sum_{s' \in Sat(\Phi)} \widetilde{\pi}(s, s', t) \cdot \widetilde{\pi}^{(N)}(s', Sat(\Phi)) > p \qquad \Rightarrow \qquad \pi(s, Sat(\Phi)) > p
$$

$$
\text{(b)} \sum_{s' \in Sat(\Phi)} \widetilde{\pi}(s, s', t) \cdot \widetilde{\pi}^{(N)}(s', Sat(\neg\Phi)) > 1-p \qquad \Rightarrow \qquad \pi(s, Sat(\Phi)) < p
$$

Note that the a priori error that is chosen for the initial distribution does not influence the above inequalities as the computed probability is always an underestimation. Thus, in case the underestimation is already larger than p or $1-p$, we can be sure that the actual probability is as well. The a priori error of the initial distribution should be chosen reasonably small, in order to take enough probability mass into account, otherwise the iteration might not stop.

Given $\neg\Phi \vee \Psi$ is independent as of $\overline{l}$ and the number of steps considered by uniformization for an a priori error $\varepsilon_{\nu,t}$ is n. Then, in case the above iteration stops after N steps for all starting states, front $R(\overline{l}+\overline{n}+\overline{N})$ is representative for all fronts larger $\overline{l}+\overline{n}+\overline{N}$.

4.4.5 Proof of Proposition 9 for the Until operator

The proof of Proposition 9 for the Until operator when model checking JQNs follows the same lines as the proof of Proposition 5 for model checking QBDs (Section 3.4.6).

We want to show that for any until formula $\mathcal{P}_{\bowtie p}(\Phi\mathcal{U}^I\Psi)$ there exists a independence vector $\overline{g}$, under the assumption that for no state s the probability measure is exactly equal to p, hence, $Prob(\mathsf{s}, \Phi\mathcal{U}^I\Psi) \neq p$.

We do this for time-bounded until formulas $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{[0,t]}\ \Psi)$ in more detail. Assume that Φ is independent as of $\overline{g}_1$ and Ψ is independent as of $\overline{g}_2$. The entry-wise maximum of $\overline{g}_1$ and $\overline{g}_2$ is then denoted $\overline{g}'$. The termination criterion (cf. Section 3.4.1) of uniformization with representatives ensures that for every state $\mathsf{r} \in R(\overline{g}')$ there exists an $N_\mathsf{r} \geq 1$ such that either

$$\forall \mathsf{s} \in S_\mathsf{r} : \gamma_\mathsf{s}(t) > p \text{ or } \forall \mathsf{s} \in S_\mathsf{r} : \gamma_\mathsf{s}(t) < p.$$

The maximum of all N_r is then the new independence vector $\overline{g}$ for independence of the formula $\mathcal{P}_{\bowtie p}(\Phi\ \mathcal{U}^{[0,t]}\ \Psi)$. However, we still have to discuss that the algorithm always terminates. Assume that it does not do so, then

$$\forall \mathsf{s} \in \mathsf{S} : \lim_{n\to\infty} \gamma_\mathsf{s}^{(n)}(t) < p \quad \textit{and} \quad \forall \mathsf{s} \in \mathsf{s} : \lim_{n\to\infty} \left(\gamma_\mathsf{s}^{(n)}(t) + \underbrace{\varepsilon_{t,\nu}^{(n)}}_{\to 0} \right) > p.$$

But since $\lim_{n\to\infty} \varepsilon_{t,\nu}^{(n)} = 0$, we obtain

$$p < \gamma_\mathsf{s}(t) = \lim_{n\to\infty} \gamma_\mathsf{s}^{(n)}(t) < p,$$

which is not possible. Consequently the algorithm will always stop, thereby having computed an N_i for each representative state, and so the corresponding until formula is independent as of independence vector $\overline{\max_i(N_i)}$. For the unbounded until formula, independence as of $\overline{g}$ can be proven similarly. □

4.5 Case study: An e-business site

Modeling an e-business site as Jackson queueing network facilitates analyzing its scalability. This is extremely important as customers become dissatisfied easily in case such a site is overloaded. We are able to model an e-business site in as much detail as shown in [59], however, we use a model with one queue per server instead of two, to keep the model concise. On the other hand, where [59] only analyzes average response times, we are able to analyze a wide range of more advanced measures, given by the logic CSL and the new analysis algorithm.

The case study is further organized as follows: Section 4.5.1 describes the system and introduces the model, in Section 4.5.2 the steady-state behavior of the system is analyzed, whereas in Section 4.5.3 we show how the dynamic termination criterion improves the performance of the transient analysis. The tool usage of this case study is discussed in Section 4.5.4.

4.5.1 System description and model

Consider an online retail shop, where requests arrive from a potentially infinite customer base. The site itself consists of three servers: a web server, an application server and a database server. The requests are first dealt with by the web server that manages all the web pages and handles the direct interactions with the customer. The application server implements the core logic of the site and the database server stores persistent information about registered customers, prices and article descriptions.

Arriving requests first visit the web server, after which they are either forwarded to the application server, routed back to the web server itself or leave the system, when they have been completed. Jobs that visit the application server are either forwarded to the database server or routed back to either the web server or the application server. From the database server, jobs are routed to either the application server or back to the database server itself. Note that requests can only leave the system via the web server. As illustrated in Figure 4.10, the associated JQN then consists of three unbounded queues modeling the buffer of the web server, the buffer of the application server and the buffer of the database server, respectively.

Requests from the infinite population arrive according to a Poisson process with rate λ and are then routed according to the routing matrix

$$\mathbf{R} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.4 & 0.3 & 0.3 & 0 \\ 0 & 0.3 & 0.4 & 0.3 \\ 0 & 0 & 0.7 & 0.3 \end{pmatrix}. \tag{4.19}$$

To compute the steady-state probabilities we need the arrival rates per queue. They are computed by solving the traffic equations (4.20) which follow directly from Fig-

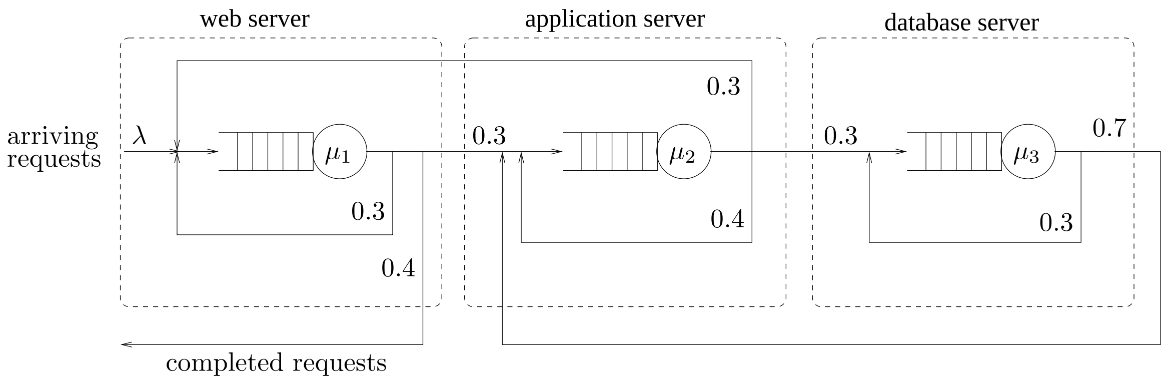


Figure 4.10: Queueing network model for an online auction site

ure 4.10:

$$\begin{aligned} &\lambda_1 = \lambda + 0.3 \cdot \lambda_2 + 0.3 \cdot \lambda_1 \\ \wedge\ &\lambda_2 = 0.3 \cdot \lambda_1 + 0.4 \cdot \lambda_2 + 0.7 \cdot \lambda_3 \\ \wedge\ &\lambda_3 = 0.3 \cdot \lambda_2 + 0.3 \cdot \lambda_3. \end{aligned} \tag{4.20}$$

The arrival rates and the service rates per queue are given in Table 4.1. To conduct

parameter	λ_1	λ_2	λ_3	μ_1	μ_2	μ_3
sec^{-1}	$\frac{5}{2} \cdot \lambda$	$\frac{5}{2} \cdot \lambda$	$\frac{15}{14} \cdot \lambda$	5	5	3

Table 4.1: Numerical values for the parameters of the model

steady-state analysis we also require the system to be stable. From

$$\rho_1 = \frac{\lambda_1}{\mu_1} < 1 \ \wedge\ \rho_2 = \frac{\lambda_2}{\mu_2} < 1 \ \wedge\ \rho_3 = \frac{\lambda_3}{\mu_3} < 1, \tag{4.21}$$

it follows directly that $\lambda < 2$.

To analyze the scalability of the e-business site, we define the CSL formula `overflow` to indicate that all queues are filled above a certain threshold as

$$\texttt{overflow} = (s_1 \geq \mathsf{full}) \vee (s_2 \geq \mathsf{full}) \vee (s_3 \geq \mathsf{full}),$$

for different possible values of full. The atomic proposition

$$\texttt{no\_overflow} = (s_1 < \mathsf{full}) \wedge (s_2 < \mathsf{full}) \wedge (s_3 < \mathsf{full}) = \neg\texttt{overflow}$$

indicates that all queues contain less than full requests.

4.5.2 Model checking steady-state properties

We want to analyze for which combinations of the parameters full and λ the steady-state probability to be in `overflow` state is reasonably small. Stated in CSL, we want to analyze $\mathcal{S}_{<p}(\texttt{overflow})$ for different probability bounds p.

Figure 4.11 shows the steady-state probability to be in an `overflow` state for $\lambda \in [0, 2[$ for $\mathsf{full} \in \{5, 10, 15, 20, 25, 30\}$. As can be observed for increasing values of parameter full the steady-state probability stays below a given threshold for larger values of λ, however, the slope becomes more steep.

For the administrator of an e-business site it is of interest to know the maximum possible combination of λ and full for which the steady-state probability is still below a given threshold. Figure 4.12 shows the maximum value of λ that can be accommodated such that $Sat(\mathcal{S}_{<p}(\texttt{overflow})) = \mathsf{S}$ holds for different probability bounds $p \in \{0.05, 0.1, 0.2, 0.3\}$. For small values of full the slope is quite steep and for higher values of full it flattens. This phenomenon is well-known as the *law of diminishing return*, meaning that if the allowed buffer occupancy is increased above a certain point, the throughput increases at a decreasing rate.

Figure 4.12 shows that, for probability bound $p = 0.1$, if the buffer capacity is increased by 100 percent (from 20 to 40) the allowed λ is only increased by 12 percent from 1.6 to 1.8. At approximately $\mathsf{full} = 6$, the slope decreases. This point is called point of diminishing return, at which the derivative is at its maximum.

4.5.3 Model checking time-bounded until

Figure 4.13 shows the number of uniformization steps needed for model checking

$$Sat(\mathcal{P}_{\geq p}(\texttt{overflow}\ \mathcal{U}^{[0,t]}\ \texttt{no\_overflow})) \text{ for } t = \{5; 10; 5\},$$

depending on the probability bound p. We show the number of iterations with the dynamic termination criterion, as well as the a priori computed number of steps required for an error $\varepsilon_{t,\nu}^{(n)} = 10^{-7}$. Clearly, the a priori number of steps is independent of the probability bound p and increases with time bound t.

After 0 steps the comparison can be evaluated for $p = 0$ for all time bounds when using the dynamic termination criterion. Then the number of iterations first increases steeply and the maximum number of iterations is reached for a probability bound at most 0.2 for all four time bounds. In general, the number of iteration steps using the dynamic termination criterion decreases for larger p. Note that the step size of p, as shown in Figure 4.13 was taken to be 0.01.

The number of iterations in Figure 4.13 clearly varies over time. A peak occurs whenever the computed probability for some state is really close to the probability bound p we have to compare with. The maximum number of iterations with the dynamic termination criterion approximately equals the a priori computed number

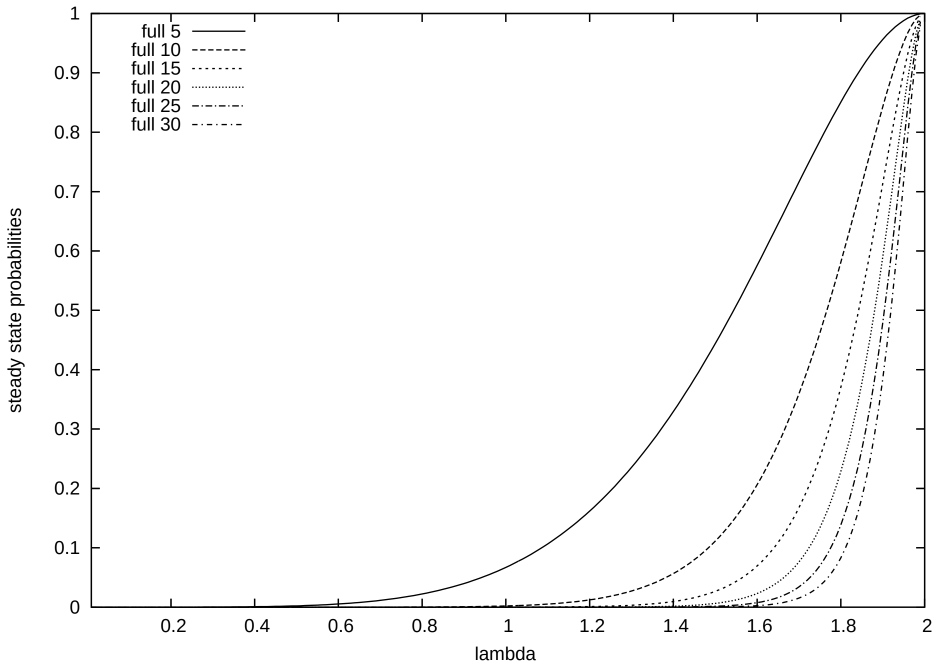


Figure 4.11: Steady-state probability to be in a `overflow` state

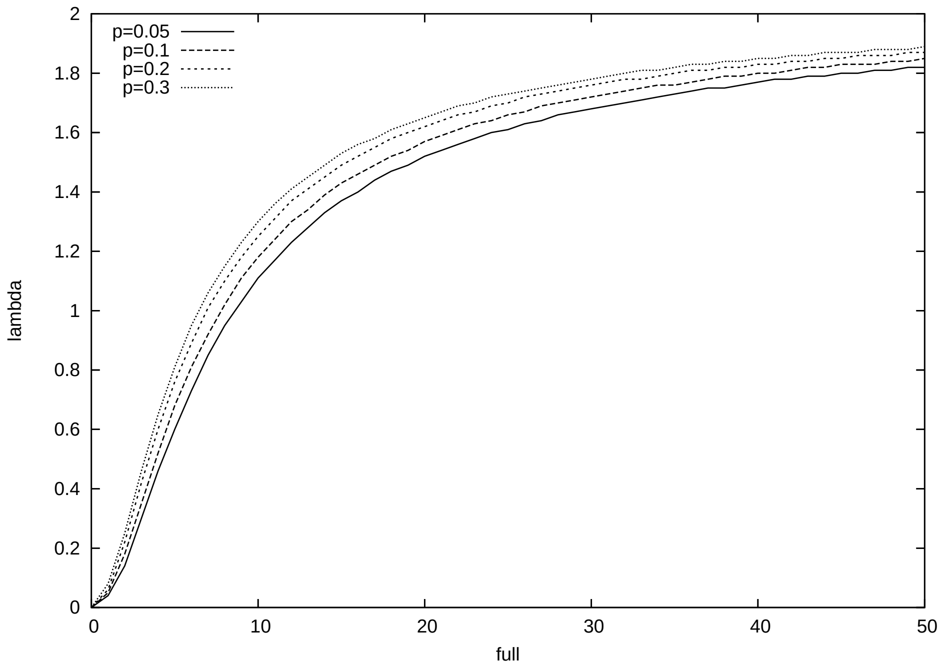


Figure 4.12: Maximum λ for which $\mathtt{s} \models \mathcal{S}_{<p}(\mathtt{overflow})$

of steps for $\varepsilon_{t,\nu}^{(n)} = 1 \cdot 10^{-7}$. For larger time bounds t, the difference between the number of iterations for the dynamic and the a priori termination criterion increases, showing the efficiency gain using the dynamic termination criterion.

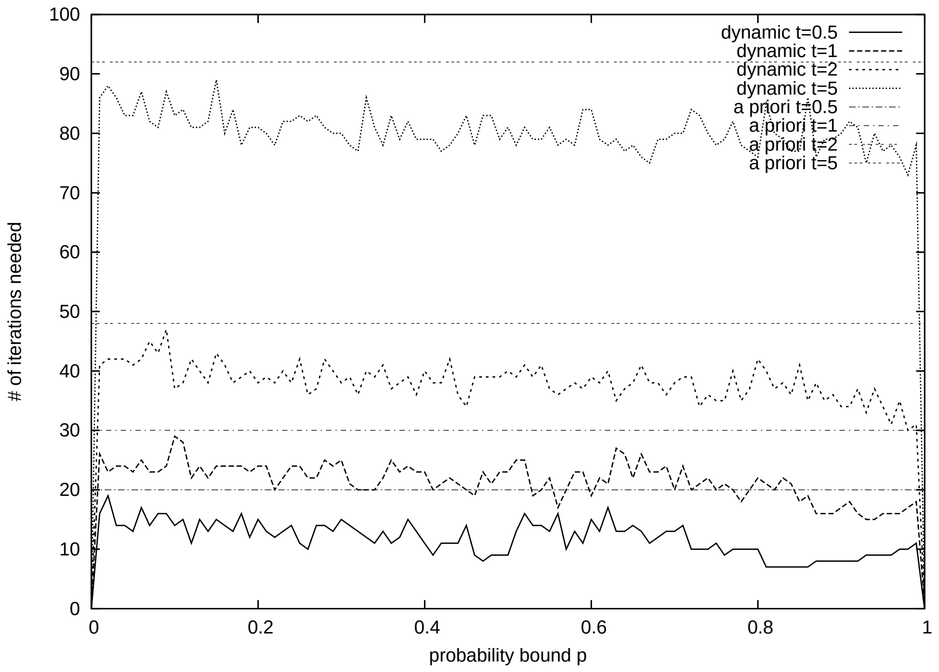


Figure 4.13: Number of iterations needed for model checking $\mathtt{s} \models \mathcal{P}_{\geq p}(\mathtt{overflow}\ \mathcal{U}^{[0,t]}\ \mathtt{no\_overflow})$ with the dynamic termination criterion and with a priori error

In Table 4.2 the first group of rows shows the minimum and maximum number of iterations per time bound with the dynamic termination criterion and the a priori computed number of iterations per time bound. The second group of rows then show the finite number of states that is considered of the underlying infinite Markov chain $\mathcal{J}$, depending on the number of iterations and again depending on the time bound t. The corresponding number of states in the absorbing Markov chain $\mathcal{J}[\neg\Phi \vee \Psi]$ is shown in the third row. In the last group of rows, the numerical error $\varepsilon_{t,\nu}^{(n)}$ for the corresponding number of iterations is given.

Using the dynamic termination criterion, the number of iterations that is necessary to decide whether $\mathtt{s} \models \mathcal{P}_{\geq p}(\mathtt{overflow}\ \mathcal{U}^{[0,t]}\ \mathtt{no\_overflow})$ for all $s \in \mathsf{S}$ grows with increasing time bound t. This is due to the fact, that with a larger time bound more steps can be taken. With an increasing number of iterations also the considered finite part of the underlying infinite Markov chain grows. In contrast, with more iterations, the introduced numerical error $\varepsilon_{t,\nu}^{(n)}$ decreases. Therefore, the error bound in column *dynamic min* is larger than the error bound in column *dynamic max*. For time bound $t = 5$, $\varepsilon_{t,\nu}^{(n)} = 1.1 \cdot 10^{-7}$ is enough to decide

	t	**dynamic**		**a priori**
		min	max	$\varepsilon_{t,\nu}^{(n)} = 1 \cdot 10^{-6}$
number of iterations	0.5	7	19	20
	1	15	29	30
	2	30	47	48
	5	73	89	92
number of states in $\mathcal{J}$	0.5	7770	260130	273819
	1	23426	877975	1004731
	2	91881	2081156	2362041
	5	782246	1333300	1456935
number of states in $\mathcal{J}[\neg\Phi \vee \Psi]$	0.5	121768	252132	265821
	1	532276	869977	996733
	2	1406912	2073158	2354043
	5	782246	1325302	1448937
uniformization error $\varepsilon_{t,\nu}^{(n)}$	0.5	$1.3 \cdot 10^{-1}$	$4.0 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
	1	$2.4 \cdot 10^{-2}$	$3.1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
	2	$1.5 \cdot 10^{-2}$	$1.0 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
	5	$1.1 \cdot 10^{-3}$	$3.45 \cdot 10^{-7}$	$1 \cdot 10^{-7}$

Table 4.2: Numerical values for the parameters of the model

that $\texttt{s} \models \mathcal{P}_{\geq p}(\texttt{overflow}\ \ \mathcal{U}^{[0,t]}\ \ \texttt{no\_overflow})$ for probability bound $p = 0.98$. Whereas $\varepsilon_{t,\nu}^{(n)} = 3.45 \cdot 10^{-7}$ is enough to decide this for all probability bounds $p \in \{0.0, 0.01, 0.02, \ldots, 0.99, 1.0\}$. However, this small error is only necessary to decide the validity of the CSL formula for probability bound $p = 0.15$. Note that the given number of iterations and the given error might not be enough to decide for every other probability bound.

The last column of Table 4.2 shows the number of iterations that has to be taken to keep $\varepsilon_{t,\nu}^{(n)} \leq 1 \cdot 10^{-7}$. Figure 4.13 shows that an error of $1 \cdot 10^{-7}$ is always enough to decide whether $\texttt{s} \models \mathcal{P}_{\geq p}(\texttt{overflow}\ \ \mathcal{U}^{[0,t]}\ \ \texttt{no\_overflow})$ for all $s \in \mathsf{S}$. The number of states of the infinite Markov chain that has to be considered is always slightly larger than for the maximum in case the dynamic termination criterion is used.

4.5.4 Tool usage

To compute the steady-state probabilities for this case study, the algorithm presented in Section 4.2.6 has been coded in C++. The computation of the steady-state probabilities for 200 different values of λ is done within 0.2 seconds on an Intel Pentium 4 with 3 GHz and 1 GB main memory.

To model check the time bounded until operator the JQN has been transformed manually into a stochastic Petri net [19]. To model the possible infinite population of the JQN, an additional place *finite* has been added to the SPN from which all arrivals take place and to which all departures are routed. In case the inner formula $\neg\Phi \vee \Psi$ is independent of g and given the number of iterations is n, the place *finite* is initialized with $g+2\cdot n$ tokens to account for the $g+n$ possible starting fronts and the $g+2n$ possible goal fronts. Then the CSPL implementation by Bell [12] is used to generate the underlying Markov chain and an implementation of the uniformization method for finite state Markov chains by Cloth [22] is used to compute the transient probabilities. A script emulated the dynamic behavior of the algorithm. The time to compute the transient probabilities ranges from 0.4 seconds to 22 seconds for the different time bounds, when using the dynamic termination criterion, and between 1.3 seconds and 26 seconds, when using the a priori termination criterion.

4.6 QBDs versus JQNs

When comparing the algorithms for CSL model checking of QBDs and the algorithms for CSL model checking of LQNs, we see that the steady-state operator is checked with a specific algorithm for each model class. The steady-state operator on QBDs is checked with an iterative method based on the matrix geometric method and a termination criterion that is especially tailored for this algorithm. For JQNs, the steady-state operator can be checked directly via a closed-form solution that is derived from Jackson's theorem.

For model checking the next operator and the until operator with its different time bounds intervals, however, we have proposed approaches similar to those for finite CTMCs. The algorithms are then tailored specifically to QBDs and JQNs. The only difference in the algorithms corresponds to the different partitioning of the infinite-state space for QBDs and JQNs:

- For model checking the until operator for QBDs, the state space is split in levels. This is reflected in Equations (3.2 – 3.6) in the second summation, that ranges over all $Sat^i(\Phi)$.
- For model checking the until operator for JQNs the state space is split in fronts. This is reflected in Equations (4.10 – 4.14) in the second summation, that ranges over all $Sat^{F(\bar{\imath})}$.

Basically, the algorithms as presented in Section 3.2 and in Section 4.2 can be used on every class of infinite-state Markov chain that has a structured, repetitive state space, which can be infinite in several dimension, that has constant transition rates, and atomic propositions that do not change any more from some point on in the infinite dimensions. For such state space, a partitioning

$$\mathsf{S} = \bigcup_{i=0}^{\infty} F(\bar{\imath}) \text{ with } F(\bar{\imath}) \cap F(\bar{\jmath}) = \varnothing \text{ for } i \neq j,$$

comparable to the general partitioning for JQNs, as presented in Section 4.1.4, can be defined. The definition of the partitioning highly depends on the state space structure. The requirement that the state space is structured and repetitive, ensures that we can use the concept of independence and representatives as for QBDS and JQNS. However, note that the complexity of model checking such a class of infinite CTMCs also highly depends on the structure of the state space.

4.7 Summary

In this chapter we presented model checking algorithms for checking CSL properties for a very general class of queueing networks, namely for labeled Jackson queueing networks. The underlying state space of an JQNs is a highly structured CTMC, that is, of infinite size in as many dimensions as there are queues. We introduce a new notion of property independence on JQNs that is needed for model checking. Steady-state probabilities are computed in (stable) JQNs with well-known product-form results and transient probabilities are computed with a a new uniformization-based approach. We provided a running example throughout the paper to illustrate our approach.

Chapter 5

Beyond CSL model checking QBDs and JQNs

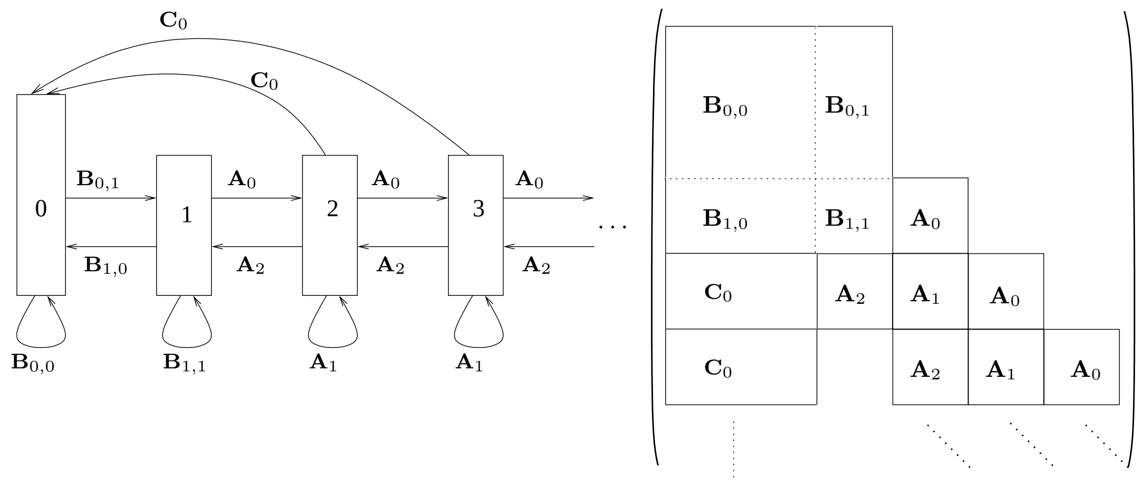
In the previous two chapters we have developed CSL model checking algorithms for QBDs and JQNs. In this chapter, we describe extensions of QBDs in Section 5.1 and discuss whether model checking these extensions with the framework proposed earlier is feasible [66]. In Section 5.2 we present several extensions for JQNs and explain whether model checking these extensions fit into our frame work. Section 5.3 introduces a reward structure and presents CSRL model checking algorithms for QBDs and JQNs. Then related work on transient analysis of infinite CTMCs and on model checking infinite Markov chains is presented in Section 5.4. We conclude this chapter with a summary in Section 5.5.

5.1 Model extensions for QBDs

We discuss model checking QBDs with *resets* in Section 5.1.1 and model checking QBDs with *batch arrivals and/or service* in Section 5.1.2. In Section 5.1.3 we introduce QBDs with *periodic atomic propositions* and explain that they can be model checked with our approach as well. Section 5.1.4 presents *tree-like* QBDs and explains why model checking this class of infinite CTMCs with our approach does not scale well.

5.1.1 Resets

In standard QBDs, transitions can only occur between states of the same level or between states of neighboring levels. In QBDs with *resets* we additionally allow for transitions that lead from any state in any repeating level to the boundary level, as shown in Figure 5.1(a). This is useful to model situations where all jobs in a system are lost due to some special event (like a server breakdown). The generator matrix



(a) transitions in a QBD with resets (b) generator matrix

Figure 5.1: QBD with resets

as in Figure 5.1(b) is then composed out of nine matrices, where $\mathbf{C}_0$ denotes the transitions (resets) from the repeating levels to the boundary level. Note that the reset transition matrix $\mathbf{C}_0$ is the same for all repeating levels.

Again, considering level-independent atomic propositions, CSL formulas on QBDs with buffer resets are *not* level-independent in general. Even though it is now possible to reach the boundary level from every repeating level in one step, there still is the possibility to reach the boundary level via repeating levels. Due to this, the transient probabilities to reach the boundary level differ from level to level and, hence, we again have to apply the concept of level independence as of level k.

The steady-state probabilities can be computed with known techniques [74, 56, 61]; the only difference to applying matrix-geometric methods on standard QBDs is that the boundary equations (cf. Appendix A) must now account for the new matrix $\mathbf{C}_0$, yielding $v_0 = v_0\mathbf{B}_{0,0} + \sum_{j=1}^{\infty} v_j\mathbf{C}_0$, with $\pi = (v_0, v_1, v_2, \cdots)$ being the steady-state vector. As $\mathbf{C}_0$ is a constant matrix, this equation can easily be solved due to the geometric structure of the solution vectors v_j.

Computing the transient probabilities can again be done with uniformization. As we require the transition matrix $\mathbf{C}_0$ to be constant, we still have a finite number of different diagonal entries so that the uniformization rate can be determined. Model checking the until operator on QBDs with resets thus has the same complexity as on standard QBDs. Note that the level diameter can still be computed as addressed in Chapter 3. In conclusion, model checking of QBDs with buffer resets can be done exactly as for standard QBDs, as presented in Chapter 3.

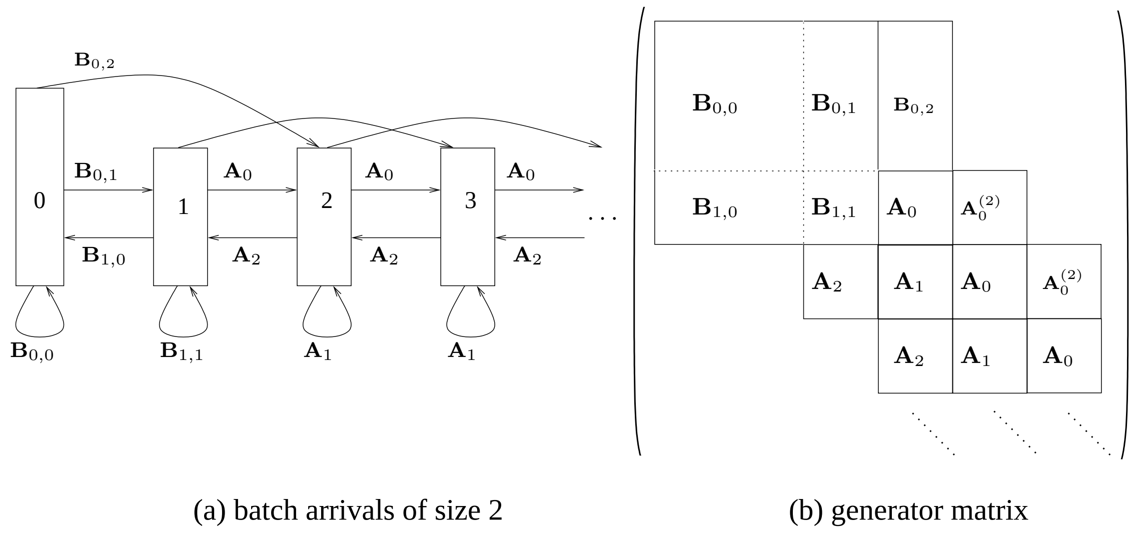


Figure 5.2: QBD with batch arrivals

5.1.2 Batches

For modeling the case that jobs enter or leave a system not only in single instances but also in *finite* batches, we need transitions between non-neighboring levels. We distinguish between (a) *batch arrivals*, (b) *batch departures*, and (c) the *general case*, which contains (a) and (b). The batches can have different size distributions. To describe a QBD with batch arrivals we need additional transition matrices $\mathbf{A}_0^{(i)}$, for every possible batch size i. $\mathbf{A}_0^{(1)}$ just equals $\mathbf{A}_0$. The new transition matrices for batch departures are denoted as $\mathbf{A}_2^{(i)}$. Since we allow only for finite batches, the number of extra matrices is finite as well.

The generator matrix of a QBD with batch arrivals, see Figure 5.2, has an upper block-tridiagonal form and can be seen as a special $M|G|1$ process; the generator matrix of a QBD with batch departures has lower block-tridiagonal form and can be seen as a special $G|M|1$ process. In case of both batch arrivals and departures, the generator matrix takes a block-banded form. By regrouping as many states into one level as necessary to guarantee that transitions entering or leaving a (new) level are restricted to neighboring levels only, we can transform the QBD with batches to a standard QBD. This procedure always works, as long as the maximum batch size is finite [40]. With such a regrouping, the level size is multiplied with the maximum batch size. If the QBD with batch arrivals or departures has level-independent atomic propositions then the regrouped QBD has level-independent atomic propositions as well, because only complete levels are regrouped.

The regrouped QBD can then be model-checked with the standard procedure. We just have to make sure that the left and right level-diameter (cf. Chapter 3), are adapted accordingly to keep uniformization efficient. As reducing QBDs with batch arrivals/departures to standard QBDs might lead to a considerable increase of the QBD order, it might be advantageous to model check these QBDs, without regrouping. This can be done with specialized matrix-geometric algorithms for $M|G|1$ and $G|M|1$ processes with batch arrivals and services [56], [74], or with the spectral expansion method [60]. Uniformization can be done as for standard QBDs, we just have to consider that one uniformization step possibly crosses more than one level. That is, the amount of reachable levels to the right and to the left grows according to the maximum batch size for arrivals and departures. Thus, for model checking the until operator and to maintain the notion of level independence, the batch sizes have to be taken into account accordingly.

5.1.3 Periodic atomic propositions

Until now we always required QBDs with level-independent atomic propositions for model checking CSL. However, we are able to check QBDs with so-called *periodic atomic propositions* as well. An atomic proposition is called periodic with period p, iff its validity repeats every p levels. Atomic propositions with period $p = 1$ are called level-independent, with period $1 < p < \infty$ are called periodic, and with period $p = \infty$ are called level-dependent. For example, an atomic proposition *odd* which is valid in states with odd level index, is periodic with period 2. QBDs with periodic atomic propositions with period p can be regrouped to QBDs with level-independent atomic propositions by combining p levels to one. In the case of several periodic atomic propositions with different periods the number of levels that need to be combined is given by the least common multiple of all periods. As far as we know, regrouping is the only possibility to model check QBDs with periodic atomic propositions.

5.1.4 Tree-like and tree-structured QBDs

We address the state space underlying tree-like and tree-structured QBDs, before we discuss the possibilities and problems when model checking such infinite-state CTMCs.

State space structure

Tree-structured QBDs arise from $PH|PH|1$ queues with *preemptive last in first out* (LIFO) service discipline. They were first introduced in 1994 by Yeung and Sengupta [87] with transitions between siblings, to parents, children and to all ancestors. The underlying stochastic process corresponds to an $M|G|1$ process. Later on different types of transitions have been proposed; in [78] transitions between siblings, to parents, children and to all descendants are considered, which corresponds to an $G|M|1$ process. Yeung and Alfa [88] allow for transitions between siblings, to parents

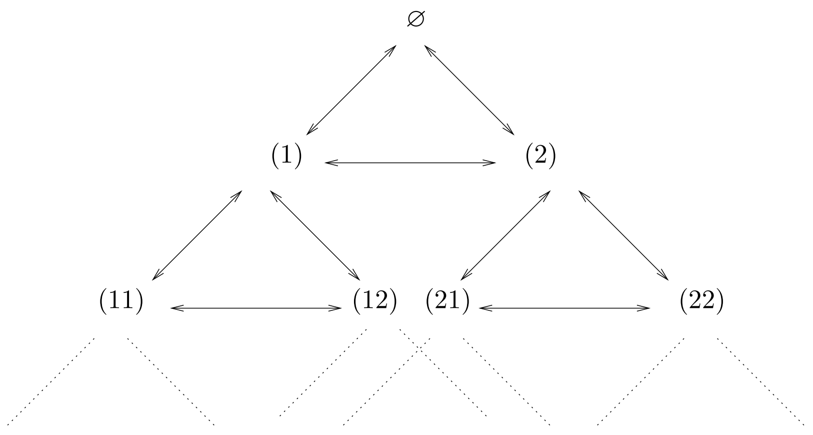


Figure 5.3: $M|PH|1$ queue with LIFO with a tree-structured state space

and children, only. They all solve the arising steady-state equation system in a matrix geometric way.

In Figure 5.3 a tree-structured QBD is shown, where each node in the tree has two different children (branching factor 2). State $\varnothing$ represents the empty queue. The branching factor equals the number of phases in the phase-type service distribution. In the empty tree-structured QBD from Figure 5.3 arriving jobs can be either in service phase 1 or in service phase 2, represented by the nodes (1) and (2). Due to the preemptive LIFO service discipline, a new job arrival causes the preemption of the job currently in service and the service phase of the preempted job needs to be stored. This results in a tree-structured state space structure, where a node of the form $(x_1, x_2, \dots, x_n)$ represents n jobs, each in service phase $x_i \in \{1, \dots, d\}$, where the jobs $1, 2 \dots, n-1$ have been preempted and job n is currently in service.

Note that with *first in first out* (FIFO) service discipline the state space is just a usual QBDs, as jobs cannot be preempted and only the service phase of the job currently in service needs to be stored.

Tree-like stochastic processes and the numerical methods to solve the steady-state equation system have been introduced by Bini et al. [15]. In contrast to tree-structured QBDs, they do not allow for transitions between siblings. A tree-like stochastic process with branching factor 2 is shown in Figure 5.4. Each node contains two states, representing the service phase of the job currently in service. The boundary node represents one job in the system and a job from layer n represents n jobs in the system. A job that is preempted moves to the queue; according to its service phase a transition is taken: if it was preempted in service phase 1, transitions $L \downarrow$ is taken and if it was preempted in service phase 2, transition $R \downarrow$ is taken to the next lower layer. When a job is resumed, transition $L \uparrow$ or transition $R \uparrow$ is taken to the next higher layer, depending on the service phase of the resumed job.

Van Houdt and Blondia show in [81] that an arbitrary tree-structured QBD can be embedded in a tree-like QBD. Therefore, in the following, we concentrate on

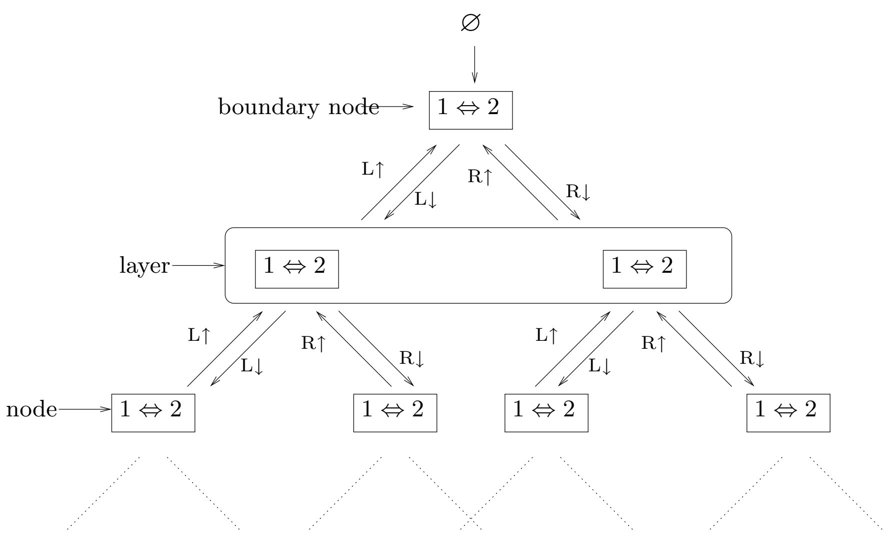


Figure 5.4: Tree-like QBD

tree-like QBDs. Literature on tree-like QBDs mostly discusses efficient algorithms to solve steady-state probabilities. However, Van Houdt and Blondia introduce an approximate approach, to perform transient analysis of tree-like processes by using marked time-epochs [85].

Model checking

For model checking purposes we require the same atomic propositions for corresponding states of all nodes, however, different atomic propositions may be valid in corresponding states of left and right children.

To check the *steady-state operator* on tree-like QBDs, the steady-state probabilities need to be computed in a recursive and layer-wise manner. Then the iterative comparison as introduced in Section 3.2.5 can be used to decide the validity of the steady-state operator. Efficient algorithms for steady-state analysis of tree-like QBDs in discrete-time exist, that can be adapted for the continuous-time case.

Model checking the *next operator* can easily be done, using the approach from model checking finite state Markov chains, that is also employed on QBDs and JQNs. On a tree-like QBD with branching factor d, we need d representative nodes, that is, d^2 representative states.

Checking the validity of an CSL *until formula* on a single starting state, we need to consider all states that can be reached for a given number of steps:

- When purely descending (moving downwards in the tree), we need to consider the $d-1$ other states in the same node and with the first step we can reach another d nodes in the next layer, where each node has in turn d states (d^2 states in total). With two steps, it is possible to reach again the next layer,

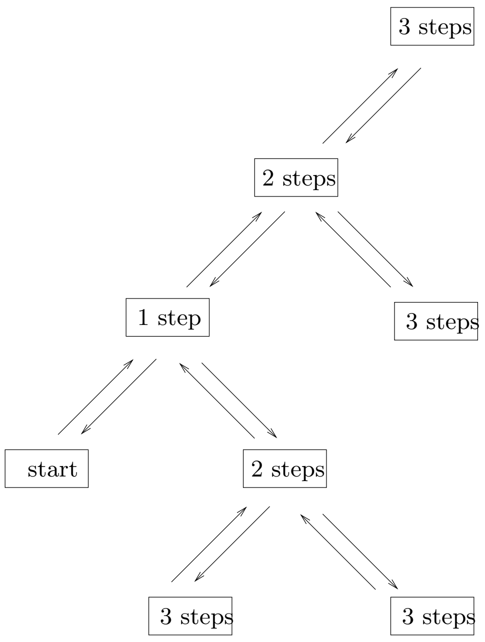


Figure 5.5: Nodes reachable per step

with its d^2 nodes (d^3 states in total). Thus, with n steps it is possible to reach

$$\sum_{i=0}^{n} d^{i+1} = d \cdot \frac{d^{n+1}-1}{d-1}$$

states when descending.

- Ascending (moving upwards in the tree) and then possibly descending again, one can reach the next upper node with every next step, as well as the $d-1$ siblings of the current node and possibly their descendents, in a recursive manner. Figure 5.5 shows that starting from the left-most node, denoted *start*, d states can be reached in the first step, another d^2 states can be reached with the second step and again d^3 states can be reached with the third step. Note that once the boundary node is reached, less states can be reached. In total this leads to a maximum of

$$\sum_{i=1}^{n} d^{i} = \frac{d^{n+1}-d}{d-1}$$

states that need to be considered with n steps. For nodes situated differently in the tree, the number of states reachable per step remains the same.

To check the validity of a CSL until formula for a given starting state with n steps, we have to consider a maximum of

$$d \cdot \frac{d^{n+1}-1}{d-1} + \frac{d^{n+1}-d}{d-1} = \frac{d^{n+1}(d+1)-2d}{d-1}.$$

This is an exponential increase, however, it can be feasible for a small branching factor d and not too large time bounds (which determine n).

For computing the possibly infinite satisfaction set for an until expression, we need the transient probabilities for all starting states and for all goal states. We can possibly use the concept of uniformization with representatives to tackle the infinite-state space. Considering the structure of tree-like QBDs, all nodes, except the boundary node have the same structure. The transition probabilities for descending are the same for all nodes, however, the transition probabilities for ascending differ for nodes of one layer, depending on whether the node is a left or right child.

Considering one step with uniformization and the fact that we do not want to reach the boundary node, we need a maximum of d representative nodes, as the transition probabilities for ascending differ. They then give the transient probabilities for all other nodes. Considering n steps, we need d^n representative nodes. This is theoretically possible, however, practically not. To overcome this, abstraction techniques, as briefly touched upon in Section 5.4.2, might help.

5.2 Model extensions for JQNs

In Section 5.2.1 we introduce Jackson queueing networks with interrelated atomic propositions and discuss the corresponding model checking procedure. The consequences of more general queueing stations for CSL model checking are discussed in Section 5.2.2.

5.2.1 Interrelated atomic propositions

Recall, that we required atomic propositions of the following form in Section 4.2.1:

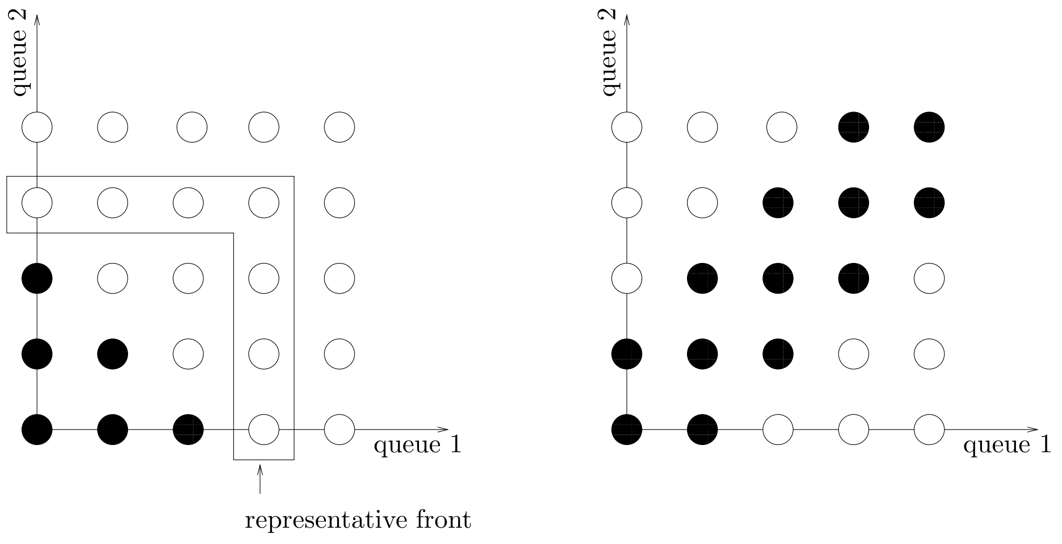
$$ap = \bigwedge_{m=1}^{M} (s_m \bigtriangleup g_m), \text{ for } g_m \in \mathbb{N} \text{ and } \bigtriangleup \in \{<, \geq\}.$$

The customers per queue are independently compared with a given per-queue threshold g_m. Several other possibilities for atomic propositions can be considered. In the following we discuss two possibilities to define interrelated atomic propositions:

1. To compare the total amount of customers in several queues with an overall threshold, atomic propositions of the form

   $$ap = \left(\sum_{m=1}^{M} s_m \cdot \mathbf{1}_{ap}(m) \right) \bigtriangleup g \text{ for } g \in \mathbb{N}, \text{ and } \bigtriangleup \in \{<, \geq\},$$

   can be defined. The indicator function $\mathbf{1}_{ap}(m)$ returns 1 if the customers of queue m are to be taken into account for the atomic proposition ap, and



(a) $s_1 + s_2 < 3$ (b) $|s_1 - s_2| < 3$

Figure 5.6: General atomic propositions for a two-queue JQN

0 otherwise. As an example for a two-queue JQN, Figure 5.6(a) the states where $s_1 + s_2 < 3$ is valid, are colored black. The underlying state space is independent as of $\overline{g} = (g, \ldots, g)$ (cf. Section 4.1.3). Clearly this is not the most tight way to split the state space, however, atomic propositions of this form can still be model checked with the algorithms presented in Section 4.2.

2. The balance between the number of customers in two queues can be expressed with atomic propositions of the form

$$ap = |s_m - s_n| \triangle g \text{ for } g \in \mathbb{N}, m, n \in \{1, \ldots, M\}, \text{ and } \triangle \in \{<, \geq\}.$$

As an example, in Figure 5.6(b) the states where $|s_1 - s_2| < 3$ is valid, are colored black. If the underlying state space is split according to Section 4.2.1, no representative front can be found such that the validity of atomic propositions does not change in the corresponding representative sets. It is left for future work to (i) find a partitioning of the state space into pairwise disjoint fronts such that with one step only the next higher and the next lower partition can be reached; and (ii), to come up with a suitable definition of representative front and representative set that facilitates model checking.

5.2.2 More general queueing stations

A queueing station is unambiguously described by the following seven properties:

1. the customer arrival process,
2. the customer service requirements,
3. the number of service providing entities,
4. the maximum number of customers in the queueing station,
5. the size of the customer population,
6. the employed scheduling strategy
7. the number of customer classes

Considering open queueing networks with infinite buffer capacity and Poisson arrivals sets the above property (1) to Poisson, and both property (4) and (5) to infinity. This leaves four parameters that can be generalized. In the following we discuss the consequences of doing so for model checking CSL formulas. A final paragraph refers to product-form solutions for computing the steady-state probabilities.

Service time distribution (with *FIFO* scheduling)

For arbitrarily distributed customer service requirements, the underlying state space is not Markovian anymore and therefore cannot be model checked with the approach presented in this thesis. The underlying state space of a queueing station with phase-type distributed service requirements is a QBD. Changing the service requirements of a single queueing station in an open queueing network from exponential to phase-type, leads to a multiplication of each state in the order of the phase-type distribution d. The number of representative states, necessary when model checking such an open queueing network where one queue has PH-type service requirements then is d times more than in case all queues have negative exponential service requirements.

Multi-server queues

In case there are m servers working independently in queueing station i, the service rates in the underlying state space differ for states $\mathsf{s} \in \{\mathsf{s}' \mid s_i < m\}$. However, for $\mathsf{s} \in \{\mathsf{s}' \mid s_i \geq m\}$, the service rate will remain the same. This enables us to use a splitting as of the maximum of $\overline{m} = (m, \ldots, m)$, and the independence vector given by the atomic propositions. Note that queueing networks where stations have infinite server capacity cannot be model checked with our framework as the service rates never stop to differ.

Scheduling

As for the employed scheduling strategy, the underlying state space of Jackson queueing networks does not change when *FIFO* or non-preemptive *LIFO* is considered. Preemptive Repeat Identical *(PRI)* scheduling forces us to store how much work is left to do on a preempted job, resulting in a non-Markovian process. Thus model checking queueing networks, employing *PRI - LIFO* with our approach is not feasible. For the same reason, model checking queueing networks, employing Round-Robin *(RR)* scheduling is not feasible. However, for the limiting case of *RR*, that is Processor Sharing *(PS)*, the underlying state space is the same as for *FIFO* scheduling. Also Preemptive Resume Different *(PRD) LIFO* does not lead to a change in the underlying state space.

In case a queueing network contains an $M|PH|1$ queue, the scheduling strategy for this queue cannot be *LIFO* as this would lead to a tree-based state space, as elaborated in Section 5.1.4.

Multi-class

A queueing network can contain different classes of customers. Following the definition in [59], a state in the underlying state space of such a multi-class network describes the distribution of customers over the different classes and the different stations in the network.

Even if *FIFO* scheduling is assumed for a single multi-class queueing station, the underlying state space takes a tree-based form as the order of jobs per queue matters. Again model checking such a state space with our approach is not practically feasible. When random scheduling is considered on a queueing station with two classes, the state space is the same as for a JQN with two queues, since only the number of jobs per class needs to be stored.

Product-form solutions

For some combinations of service time distribution and scheduling, product-form solutions for the steady-state probabilities exist, as for JQNs. In order to have a product-form solution, Trivedi [79] summarizes that queueing stations with *FIFO* scheduling have a negative exponential service time distribution, and queueing stations with a *Coxian phase type distribution* [26] may have *PS*, *LIFO-PR* or infinite server *IS* scheduling.

Baskett, Chandy, Muntz and Palacios [11] introduce four classes of multi-class queueing stations for which product-form steady-state solutions exist. The authors show that for a queueing network with queueing stations of these four classes, interconnected by a Markovian routing, product-form solutions for steady-state exist.

Even though there are no product-form solutions for queueing networks with general phase-type distributions, we can still compute the steady-state probabilities via the embedded DTMC that corresponds to the CTMC, as shown for the unbounded until in Section 3.4.4 for QBDs and in Section 4.4.3 for JQNs.

5.3 Model checking CSRL

We address syntax and semantics of CSRL in Section 5.3.1. Section 5.3.2 introduces two types of rewards for QBDs and discusses the possibilities of CSRL model checking QBDs with such rewards. In Section 5.3.3 similar reward structure are defined for JQNs.

5.3.1 Continuous stochastic reward logic

The continuous stochastic reward logic (CSRL) is a specification formalism for performability measures over CTMCs extended with a reward structure (Markov reward models (MRMs) [39]. We can also extend infinite CTMCs with a *reward structure* $\rho : S \to \mathbf{R}_{\geq 0}$ that assigns a reward $\rho(s)$ to each state s. We consider the following subset of CSRL:

Let $p \in [0, 1]$ be a real number, $\bowtie \in \{\leq, <, >, \geq\}$ a comparison operator, $I \subseteq \mathbb{R}_{\geq 0}$ a nonempty interval and AP a set of atomic propositions with $ap \in AP$, a CSRL state formula is defined as:

$$\Phi ::= ap \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{S}_{\bowtie p}(\Phi) \mid \mathcal{P}_{\bowtie p}(\phi),$$

where ϕ is a path formula constructed by

$$\phi ::= \mathcal{X}^{\leq t}_{\leq r}\Phi \mid \Phi\, \mathcal{U}^{\leq t}_{\leq r}\, \Phi.$$

The key difference to CSL is that the path-operators are equipped with two parameters. The additional parameter r represents a bound on the accumulated reward.

1. The until operator without time- nor reward constraint $\Phi\,\mathcal{U}\Psi$ corresponds to the unbounded until operator from CSL and can be checked as described in Section 3.4.4 for QBDs and in Section 4.4.3 for JQNs.

2. The until operator with only a time constraint $\Phi\,\mathcal{U}^{\leq t}\Psi$ is just the usual CSL until and can be checked as described in Section 3.4.1 for QBDs and in Section 4.4.1 for JQNs.

3. For checking the until operator with only a reward constraint, i.e., $\Phi\,\mathcal{U}_{\leq r}\Psi$, for finite CTMCs the *Duality Theorem* [7] can be used. This theorem states that the progress of time can be regarded as the earning of reward and vice versa. Formulas with only a reward constraint can then be checked as formulas with just a time constraint on a transformed CTMCS.

4. For the case with both time and reward constraint, $\Phi\,\mathcal{U}^{\leq t}_{\leq r}\Psi$, we consider how to check the formula for only one starting state. In doing so, we can use well-known algorithms as only a finite number of steps is considered on the

uniformized or discretized variant of the MRM. In order to compute the satisfaction set we have to distinguish between different reward types, as presented below.

How to solve CSRL until formulas with general intervals for time and rewards on finite Markov chains is discussed in [21].

5.3.2 Model checking CSRL on QBDs

For QBDs, we will consider three different types of rewards: *level-independent*, *level-dependent* and *periodic* rewards (see below). To model check the until operator with time and reward constraint, the following holds:

- In the case of *level-dependent rewards* we require the rewards to be an increasing function of the level-index. The satisfaction set will then always be finite, as from a certain level onwards the reward constraint cannot be fulfilled anymore. This is the case for level j, when the states of the leftmost reachable level (reachable in the sense of the maximum number of steps taken in uniformization) have a reward r_{low}, such that $r_{low} \cdot t > r$. The number of states with level index smaller than j is finite, which allows for a direct verification.
- In the case of *level-independent rewards* we require the same reward for corresponding states in different levels. In that case, the satisfaction set is potentially of infinite size. Fortunately, we will eventually find a level from which onwards the validity of the formula will be the same in all corresponding states of the repeating levels. This is just a straightforward extension of the ideas presented in Chapter 3, that can be used because of the special reward structure.
- In the case of *periodic rewards with reward period p*, the QBD with periodic reward structure can be transformed to a QBD with level-independent rewards by regrouping of levels, as discussed previously in the context of periodic atomic propositions.

To model check the until operator with just a reward bound, the Duality Theorem [7] is applicable only in case of level-independent rewards. As the transition rates are rescaled by the reward rates, the QBD structure would be destroyed otherwise. For level-independent rewards the QBD structure does not change by this transformation and the QBD can be checked as stated in Chapter 3.

There are several extensions of CSRL [7]; here we will discuss how to apply the *expected reward* ($\mathcal{E}_{\leq r}(\Phi)$) and the *instantaneous reward* ($\mathcal{E}^t_{\leq r}(\Phi)$) operator on QBDs. The semantics of the expected reward operator is:

$$s \models \mathcal{E}_{\leq r}(\Phi) \quad \text{iff} \sum_{s' \in Sat(\Phi)} \pi(s,s')\rho(s') \leq r,$$

where $\pi(s, s')$ is the steady-state probability to be in state s' when having started in state s. In case of a finite satisfaction set $Sat(\Phi)$, we have a possibly large but finite summation, that can be dealt with. For level-independent rewards and an infinite satisfaction set $Sat(\Phi)$, we do not know how to check the expected reward operator. The iterative approach that has been used in Chapter 3 to check the steady-state operator cannot be used as the probability mass is multiplied with the reward. In case the reward equals the level-index we can derive a closed-form solution for the expected reward (by applying a geometric argument to the infinite sum), hence, model checking seems feasible. Since the steady-state probabilities in a QBD are independent of the starting state, we immediately know the satisfaction set after checking the reward operator for one starting state.

The semantics of the instantaneous reward operator is:

$$s \models \mathcal{E}^t_{\leq r}(\Phi) \quad \text{iff} \sum_{s' \in Sat(\Phi)} \pi(s, s', t)\rho(s') \leq r,$$

where $\pi(s, s', t)$ is the transient probability to reach state s' from state s in time t. To calculate the transient probabilities in a QBD we always consider only a finite number of steps. That is, the instantaneous reward operator can always be checked for a single starting state s, regardless of the reward structure. To calculate the satisfaction set we distinguish between level-independent and level-dependent rewards. We will eventually find a level from which onwards the transient probabilities do not change anymore. With level-independent rewards the validity of the instantaneous reward operator does not change anymore from this level onwards. We do not know how to check the instantaneous reward operator with level-dependent rewards and an infinite satisfaction set in all cases because the reward is multiplied with the transient probabilities.

5.3.3 JQNs with a reward structure

We consider the following two types of rewards on JQNs:

- In the case of *rewards that depend on the number of jobs per queue*, we require the reward to be a function, that increases with the number of jobs in every queue. This corresponds to QBDs with level-dependent rewards, where we required the reward to be an increasing function of the level-index. As we require the reward function to increase in every queue, the same arguments as for CSRL model checking QBDs with level-dependent rewards apply and the satisfaction set will always be finite.

- In the case of *independent rewards* we require similar independency as for atomic propositions. In that case, we can find a representative front, such that all states in one representative set have the same reward. The satisfaction set is then potentially of infinite size. This corresponds to QBDs with level-independent rewards, where a representative level can be found from which onwards the validity of the formula remains the same. Similarly on JQNs, we can find a representative front, from which onwards the validity of the formula remains the same. The same arguments as for CSRL model checking QBDs with level-independent rewards applies.

5.4 Related work

We present related work on the transient analysis of infinite- state Markov chains in Section 5.4.1 and related work on model checking infinite-state Markov chains in Section 5.4.2.

5.4.1 Transient analysis of infinite-state Markov chains

As for the steady-state and transient analysis of infinite-state Markov chains, some work has been done in the past, however, not in the context as we need it for CSL model checking. We refer to the seminal work by Neuts [61] on matrix-geometric solutions for computing the steady-state probabilities in infinite-state quasi-birth-death processes. In particular, we employ the logarithmic reduction algorithm as proposed in [55].

For transient-state probabilities, there is much less work available. Zhang et al. [89] describe a Laplace-transform based technique to obtain these probabilities for QBDs, however, they do not provide the required back transformation. In his Ph.D. thesis, Van Moorsel hints at an approach called dynamic uniformization [84], and so does Grassmann [34, 33], as a technique to evaluate systems with infinite state spaces. In their well-known 1984 paper [36], Gross and Miller already refer to a possible use of uniformization for infinite-state systems. These three papers have in common that hints towards evaluating transient-state probabilities in infinite-state systems are given, but that no true algorithms or data structures are presented. Furthermore, the issue of having an infinite number of possible starting states is not addressed.

With step-wise uniformization [19], it is possible to calculate transient-state probabilities in large or even infinite-size CTMCs and DTMCs. This is done by step-wisely extending the considered state space, i.e., on-the-fly while generating the state space. Interestingly, this approach bears resemblance with the probabilistic reachability algorithm presented in [45], however, it requires (as does [45]) an unique starting state. Recently, Van Houdt and Blondia [82, 83] addressed the transient analysis of QBDs, however, in a discrete-time setting, and using an approximation technique with unknown a priori error bounds. In [86], they use their framework for a simultaneous transient analysis for all initial states.

Transient analysis on queueing networks is mostly restricted to finite state spaces. Harrison [37] presents an iterative method to solve the time-dependent Kolmogorov equations of finite queueing networks. In [17] Buchholz applies uniformization to hierarchical queueing networks that have a finite structured state space.

The way we compute the transient probabilities comes closest to the work by Le Ny and Sericola [57], in which they compute the transient queue length distribution in the very specific context of an BMAP/PH/1 queue (the equations (3–5), (10) and (11) in [57] closely resemble our recursion (3.13)). Their approach is tailored toward two very specific queuing-related measures of interest, which are less general than the transient-state probabilities we need.

In conclusion, none of the approaches available in the literature can be used for the computation of transient-state probabilities in infinite-state Markov chains for all possible starting state (or a single unique starting distribution) *and* provides error bounds. Instead, we compute all such transient-state probabilities for all possible starting states in a single computation and provide error bounds, as required for the CSL model checking procedure. Hence, none of the previously published results is directly applicable in our context.

5.4.2 Model checking infinite-state Markov chains

We are not aware of any other work that addresses the continuous-time model checking problems addressed in this thesis. There is, however, some *related* work on model-checking infinite-state systems, although none of it addresses the specific problems we address.

Work on LTL model checking for so-called *probabilistic lossy channel* systems (PLCSs) has been proposed for the evaluation of asynchronous buffer systems [13]. The main idea as presented in [6] is to reduce the LTL model checking problem to a reachability problem in a (non-probabilistic) labeled transition system. Every PLCS with just one message type and one channel with global fault semantics can be seen as a discrete time QBD, but not reversely, as not necessarily from every state in a level a transition to the next lower level can be taken in QBDs. Furthermore, every PLCS with just one message type, just one control state and global fault semantics can be seen as a degenerated discrete time JQN, but not reversely, as PLCSs do not allow for feedback between channels. The results in this area do, however, not refer to continuous time, nor do they provide model checking algorithms for full CSL. Of particular interest in this context is [45] in which the same PLCS as in [6] is addressed. This paper proposes path-enumeration algorithms to compute, with a given small error tolerance, whether another state can be reached with probability at least p. Similarly, using an iterative step-count-based state space exploration scheme, the probability for more general PCTL formulae is computed, again for individual starting states and a given error tolerance. These algorithms bear resemblance with our algorithm for the unbounded until operator. In [2] the algorithms of [45] are refined and made simpler; furthermore, some new decidability results are given. Similarly, [64] addresses also PLCS, and two very specific reachability properties for individual starting states, in an un-timed setting.

None of the above papers addressing PLCS does address continuous-time, a full-logic like CSL, or does provide a model checking algorithm for all possible starting states, as we do. By restricting to the special classes of QBDs and JQNs, we have been able to provide a richer set of model checking algorithms.

Regular model checking comprises a set of techniques for symbolic reachability analysis for parameterized and non-probabilistic infinite-state systems, based on automata theory [50, 3]. Words are used to represent states and finite-state transducers describe transitions between states. Every discrete-time QBD and JQN can be easily expressed within the regular model checking framework. As above, the results in this area do not refer to continuous time, nor do they provide model checking algorithms for full CSL.

Recursive Markov chains (RMCs) have been proposed to model probabilistic procedural programs. The main goal of model checking RMCs is to find the probability of eventually reaching a given terminating state of the RMC, starting from a given initial state. In [31] this probability is defined as the least fixed-point solution of a system of polynomial equations. Discrete-time QBDs are a subclass of RMCs, that are as expressive as tree-like QBDs. JQNs and RMCs do not overlap. Again, the RMC work does not address continuous time, nor are complete CSL model checking algorithms provided.

Finally, there is also related work on *probabilistic pushdown automata* (pPDA); as stated in [16], these models coincide with RMCs. [16] does focus on decidability results for such automata, whereas [30] presents an algorithm for evaluating the time-unbounded until operator for a single starting state only. These models do not address continuous time, nor are complete CSL model checking algorithms provided. The decidability results presented are not necessarily valid for QBDs, as QBDs comprise only a (structured) subset of pPDA.

Three-valued abstraction for continuous-time Markov chains has been introduced in [49], based on earlier work. In [32, 10] the possibly infinite CTMC is uniformized and than reduced to a so-called abstract Markov chain with upper and lower transition probability bounds. The authors show that for determining the minimum transition probability it suffices to consider only extreme schedulers. However, the number of extreme schedulers grows exponentially with the size of the state space. The method is applied to a quasi-birth-death process in [49], by abstracting corresponding states of all levels greater or equal to a given number n. In contrast to our approach this technique can be used on general infinite CTMCs; no special structure is required. However, model checking infinite CTMCs with three-valued abstraction can only be used on several starting states. It is not possible to compute the complete, possibly infinite satisfaction set. Furthermore the authors do not address the steady state operator.

Time-bounded model checking on infinite-state CTMCs is considered in [90]. The approach is based on Grassmann's dynamic uniformization [34] and is suited to model checking arbitrarily structured finite and infinite CTMCs against CSL, thereby excluding the steady-state operator. As the CTMCs may be infinite and arbitrarily structured, it is, however, impossible to compute the complete, possibly infinite satisfaction set of a given CSL formula.

5.5 Summary

In this chapter we discussed a number of extensions to QBDs and JQNs in the context of CSL model checking, as well as the extension toward CSRL model checking of QBDs with rewards. We have shown that for QBDs with resets, batches, and periodic atomic propositions, the algorithms for model checking CSL as presented in Chapter 3 still apply, after an appropriate modification of the QBD. Model checking tree-based QBDs with the same approach is not practically feasible as the number of representatives is exponential in the number of considered steps. Several extensions of JQNs have been presented as well. For multi-server and several alternative scheduling strategies, model checking with our framework is still feasible. However, for multi-class queueing networks, our approach is not practically feasible. Furthermore, we discussed how we can extend the model checking approach for QBDs toward CSRL, so that we can model check for combined performance and dependability, that is, performability measures. We discussed a number of cases for which this is possible, and we conjecture two cases for which we think CSRL model checking will not be possible for QBDs. We presented two different kinds of rewards for JQNs, such that for CSRL model checking the arguments derived for QBDs do apply. The model checking algorithms for the next operator and the until operator with its different intervals are based on the same approach for QBDs and JQNs and can be applied to every infinite Markov chain for which representatives can be found. Finally, we provided a detailed survey of related work on transient analysis of infinite CTMCs and on model checking infinite Markov chains.

Chapter 6

Bottleneck analysis for two-hop IEEE 802.11e ad hoc networks

In this chapter we perform a detailed case study on bottlenecks in two-hop ad hoc networks [69, 73], thereby using the algorithms developed in Chapter 3 and 5.

We start with a general motivation for this case study in Section 6.1, before we describe the IEEE 802.11 access mechanism and discuss the four Quality of Service (QoS) extensions in Section 6.2. The generic model for the bottleneck analysis is introduced in Section 6.3. We then describe precisely how the four IEEE 802.11e QoS-extensions are cast into this model in Section 6.4. The detailed Opnet simulation setup is discussed in Section 6.5, before we come to a careful comparison of our model's results with the simulation results in Section 6.6. Finally, the maximum throughput that can be achieved with the different QoS-extensions is presented in Section 6.7. Related work on the performance of IEEE 802.11 ad hoc networks is presented in Section 6.8 and Section 6.9 concludes this chapter.

6.1 Motivation

The availability of cheap yet powerful wireless access technology, most notably IEEE 802.11 ("wireless LAN"), has given an impulse to the development of wireless ad hoc networks. In such networks, the stations (nodes) that are in reach of each other, help each other in obtaining and maintaining connectivity. At the same time they are also competitors, as they all contend for the same resource, i.e., the shared ether as transmission medium. The medium access control of IEEE 802.11 (based on CSMA/CA) is commonly referred to as the distributed coordination function (DCF) [52, 76]. Research has shown that, effectively, the DCF tends to equally share the capacity among contending stations [14, 58]. Although this appears to be a nice fairness property, this fairness does lead to undesirable situations in case one of the nodes happens to function as a bridge toward either another group of nodes, or to the fixed internet, as illustrated in Figure 6.1.

Recently, a quality-of-service (QoS)-extension of the IEEE 802.11 standard, the so-called EDCA ("e") version has been released [1]. Roughly speaking, this extension provides mechanisms to provide preferential treatment of certain traffic classes (or nodes) over others.

We provide a modeling framework for evaluating capacity sharing strategies [71], using infinite-state Markov reward models and CSRL model checking techniques. Then, this framework is specialized towards the IEEE 802.11e protocol, including the differentiation parameters. We embed Bianchi's model and Engelstad's extensions [14, 29] into our model, to accurately describe the effective capacity, depending on the differentiation parameters in use. Furthermore, we conduct detailed simulations that show strong evidence of the correctness of the full IEEE 802.11e model and the employed techniques.

Important to stress is that our model is a flow-level model, with makes it fundamentally different from packet-level models such as [14, 29], which have been proposed to compute the share of bandwidth (radio capacity) allocated to a fixed number of sources and a bottleneck node (for various, but not all QoS enhancements). Instead, we use an extended the packet-level results [14, 29] in a (higher-level) flow-level model in which the number of active sources varies in time, depending on how quickly they are being served.

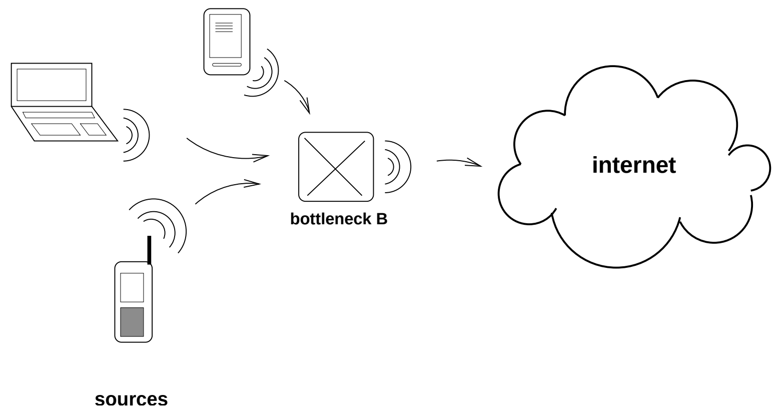


Figure 6.1: Bottleneck in a two-hop ad hoc network

6.2 IEEE 802.11 ad hoc networks

We discuss the basic IEEE 802.11 access mechanism in Section 6.2.1 before we explain how bottlenecks arise in two-hop ad hoc networks in Section 6.2.2. We present the IEEE 802.11e QoS-enhancements in Section 6.2.3.

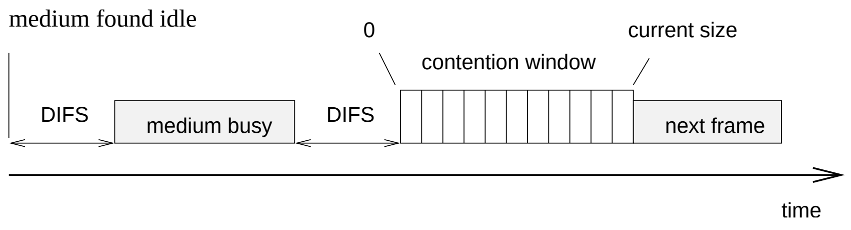


Figure 6.2: IEEE 802.11 medium access timing diagram

6.2.1 Basic operation

We address a wireless ad hoc network in which the individual nodes communicate with each other through the IEEE 802.11 protocol. In such a network the DCF organizes medium access through a carrier sense multiple access scheme with collision avoidance (CSMA/CA). All stations in such a network contend for the same radio capacity C (measured in packets per second). Whenever a station wants to send a packet, it first senses the medium until the medium is empty for at least a DIFS period (DIFS: D̲CF i̲nter f̲rame s̲pacing). If the medium is initially found empty, a station that wants medium access immediately starts transmitting after sensing the medium idle for a DIFS period. If the medium is found busy, all stations that want medium access participate in the contention mechanism. After the medium has been idle for at least a DIFS period, each station draws a random backoff time b. Each station then waits for its chosen backoff time and keeps sensing the medium. If the medium is still idle after DIFS+b time slots, the station may access the medium. This basic operation is illustrated in Figure 6.2. As a result, the station with the smallest backoff acquires medium access, if the minimum backoff is unique.

We assume that whenever two stations draw the same minimum backoff, a collision occurs. Note that in reality this depends on the relative signal strengths at the intended receiver. When the medium is sensed busy during the backoff period, the backoff is suspended and the station continues counting down the backoff after waiting the DIFS period from the moment the medium is sensed idle again. The random backoff is drawn uniformly from the so-called *contention window* (CW), initially set to $[0, \mathrm{CW}_{\min} - 1]$. For up to $r_{\max} - 1$ collisions, the size of the contention window doubles with every unsuccessful transmission and is reset to $\mathrm{CW}_{\min}$ after a successful transmission. Once the contention window has reached size $\mathrm{CW}_{\max} = 2^{r_{\max}} \cdot \mathrm{CW}_{\min}$ it stays unchanged until a successful transmission occurs. Note that in the standard IEEE 802.11 protocol, the values $\mathrm{CW}_{\min}$ and $\mathrm{CW}_{\max}$ are fixed.

6.2.2 Bottlenecks in 2-hop ad hoc networks

The scenario under study, as illustrated in Figure 6.1, has a varying number N of active nodes, the so-called sources, which are all within reach of each other. Additionally, there is one special node B, referred to as bridge or bottleneck, reachable by all sources. B is the only node that can reach the fixed internet. Thus, all traffic originating from the sources and the traffic passing through the bridge has to share the same radio transmission capacity. It has been shown that the DCF access mechanism effectively shares the radio capacity equally over all active nodes (a result of the fairness of the access mechanism itself [14, 58]). Clearly, this situation benefits the sources as a group, as they can use a relatively large share of radio capacity to send their packets, whereas the bridge becomes a bottleneck: B gets a share as any other individual node, however, it has to support the traffic of *all* other nodes. This leads to a very high buffer occupancy in B, eventually also buffer overflow, and in any case, long delays.

6.2.3 Quality of service enhancements

The Enhanced Distributed Channel Access Function (EDCAF) of IEEE 802.11e allows multiple contention instances to be simultaneously active in a single station, each supporting a certain access category (AC). Furthermore, the standard introduces four differentiation parameters (EDCA parameters), as discussed below, which can be set individually for each access category of each individual station to enable QoS provisioning [75].

We facilitate adaptive capacity sharing between stations by letting each station have a single access category, and using the EDCA parameters for differentiating between the source stations and the bottleneck station. In principle the EDCA parameters are meant for service differentiation, while we apply it here for node differentiation. Another relevant scenario for such node level differentiation is the case of UL/DL transfer in an infrastructure-based WLAN, where the access point should get a bigger share of the resources. The considered values for the differentiation parameters per access category are introduced later on in Table 6.4.

In the remainder of this paper we will analyze the following four scenarios:

0. With standard IEEE 802.11, the medium needs to be idle for at least a DIFS period before stations can start to content for medium access. After winning contention a station is allowed to send exactly one packet.

In the IEEE 802.11e QoS extension, two contention-based methods are proposed to change the above procedure:

1. The initial value of the *contention window* ($\mathrm{CW}_{\min}-1$) and/or the maximum value of the contention window ($\mathrm{CW}_{\max}-1$) are set smaller for a given station, thus, this station draws its backoff from a smaller contention window, hence, has a higher probability to win contention.

2. With so-called *arbitration inter-frame spacing* (AIFS) it is possible to assign different inter-frame spacings for different service classes (or nodes) instead of

the fixed DIFS. Thus, high-priority nodes can be assigned shorter AIFS, so that they can start counting off their backoff earlier, hence, have an advantage when contending for medium access.

A way to adapt the capacity sharing that does not alter the actual contention mechanism is the following:

3. The transmission opportunity limit ($\mathrm{TXOP}_{\mathrm{limit}}$) provides a time period during which a station may send packets after having won a contention. Thus, a station with a sufficiently high $\mathrm{TXOP}_{\mathrm{limit}}$ is able to send several packets and will thus be able to grab a larger share of the channel capacity than a station with a smaller $\mathrm{TXOP}_{\mathrm{limit}}$.

The above four parameters ($\mathrm{CW}_{\mathrm{min}}$ and $\mathrm{CW}_{\mathrm{max}}$, AIFS and $\mathrm{TXOP}_{\mathrm{limit}}$) in the IEEE 802.11e standard can be used to reallocate the amount of radio capacity given to the sources and to the bottleneck. These possibilities will be addressed in detail in the models we discuss next.

6.3 Overall capacity sharing model

We model the bottleneck B, cf. Figure 6.1, using an infinite-state stochastic Petri net (iSPN), as given in Figure 6.3. The left part of this figure contains an unbounded place (double circle) *buffer* that models the (buffer of the) bottleneck of the system. Transition *input* models the total arrival stream of packets from all active sources, whereas transition *output* models the transmission of packets leaving the bottleneck B. Note that the rates of both these transitions depend on the number of active sources and the amount of radio capacity that is allocated to each source and the bottleneck node; we come back on the form this dependency takes below. We limit the maximum number of active sources to some finite number K (taken to be 10 in most evaluations). This is a reasonable restriction, as the number of active sources in an ad hoc network cannot be arbitrarily high. We do not distinguish between individual active sources, so we can model the number of active sources as shown in the right part of the iSPN in Figure 6.3. To obtain a memoryless behavior needed for Markovian modeling, an inactive source becomes active after a negative exponentially distributed amount of time (with mean $1/\lambda$) and immediately instantiates a flow, which has a geometrically distributed length, measured in packets. The average size of a data packet is assumed to be $E[P] = 1500$ bytes, with exponentially distributed packet length. The duration of a flow does not only depend on its size but also on the radio capacity a source can use to transmit the flow. Note that the duration of a flow implicitly gives the source departure rate, as well. Following the parametric assumptions made in [80], the expected amount of work put forward per flow (the amount of packets comprising the flow) equals $E[F] = 500$ packets; the other values for the key system parameters are summarized in Table 6.1.

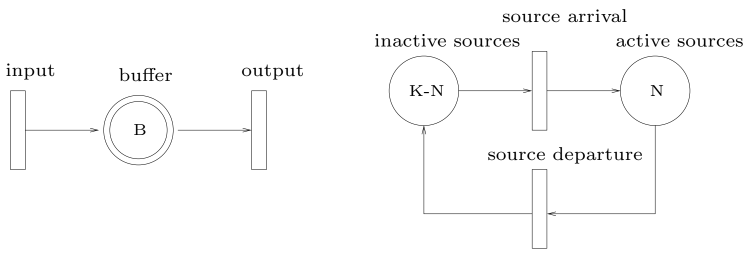


Figure 6.3: High-level model as iSPN

Table 6.1: Values for the system parameters

parameter	value
arrival rate	$\lambda \in [0.1, 0.4]\text{sec}^{-1}$
average flow size	$E[F] = 500$ packets
overall radio capacity	$C = 917$ packets/sec
maximum of active sources	$K = 10$

In Table 6.2 we list the four state-dependent transition rates of the iSPN, where N refers to the current number of active sources (i.e., the number of tokens in place *active sources*), and B to the current number of packets queued in the bottleneck (i.e., the number of tokens in place *buffer*). Note that the transitions *input* and

input:	**output:**

if $N = 0$ **then**
  return 0;
else
  return $C \cdot S_s(\cdot)$;
end if
	if $B = 0$ **then**
	  return 0;
	else
	  return $C \cdot S_b(\cdot)$;
	end if

source departure:	**source arrival:**
return $C \cdot S_s(\cdot)/E[F]$	**return** $(K - N)\lambda$

Table 6.2: State-dependent transition rates for the iSPN

output in fact make use of the same medium, hence, they have to share the available capacity; this is exactly what the IEEE 802.11[e] access mechanism is for! The functions $S_b(\cdot)$ and $S_s(\cdot)$ (for $\underline{\text{b}}$ottleneck and $\underline{\text{s}}$ource) now give the share of capacity that is allocated to the bottleneck and to all sources, respectively. Note that $S_b(\cdot)$ and $S_s(\cdot)$ depend on the number of currently active sources (N), as well as on whether or not the bottleneck has packets queued, or not (B).

In Section 6.4, we will present concrete expressions for the functions $S_b(\cdot)$ and

$S_s(\cdot)$; in doing so, we have achieved one generic model at the iSPN level, that can be specialized toward different QoS enhancements, by "plugging in" the appropriate bandwidth sharing functions $S_b(\cdot)$ and $S_s(\cdot)$.

Note that, in practice, the capacity C is not fixed, but depends on adaptive modulation, which tunes sending rates to experience link qualities, whereas we assume fixed capacity.

6.4 Modeling the QoS enhancements

In this section we present explicit expressions for the functions $S_s(\cdot)$ and $S_b(\cdot)$ that express the share of the wireless capacity that sources and the bottleneck receive, resp., for each of the QoS enhancements. We introduce the model of Bianchi [14] in Section 6.4.1 and the extensions by Engelstad et al. [29] in Section 6.4.2. The explicit computation of the relative throughput is discussed in Section 6.4.3 and the considered measures of interest are introduced in Section 6.4.4.

6.4.1 Bianchi's model

Bianchi [14] proposes an analytical evaluation of the saturation throughput, assuming ideal channel conditions for basic IEEE 802.11. This model allows us to accurately compute the throughput for a fixed number of independent stations under the assumption that they always have a packet to send.

It is assumed that each packet collides with constant and independent probability p at each transmission attempt. Under these assumptions it is possible to model the backoff stage, that is the size of the contention window, and the backoff time counter for a single station the discrete-time Markov chain as shown in Figure 6.4.

We use a discrete and integer time scale in this model, where a generic time slot is either an empty slot or a packet transmission. Thus, the time scale in the model is not directly related to the system time.

A new packet following a successful transmission starts in the first backoff stage with a randomly chosen backoff, that is somewhere in the first row of Figure 6.4. At the beginning of each slot time the backoff counter is decremented and a state change from the right to the left occurs. After an unsuccessful transmission, the packet enters the next backoff stage with a new uniformly chosen backoff, this corresponds to a transition to the next lower row of states in the model. Once the maximum backoff stage $r_{\max} = m$ is reached, the size is of the contention window is not increased with the a subsequent packet transmission (last row of the model).

In [14] the computation of a closed-form solution for the stationary distribution of the Markov chain is presented. For the state $b_{0,0}$ with zero backoff in the first backoff stage, the following steady-state probability is derived in [14, Eqn. (6)]:

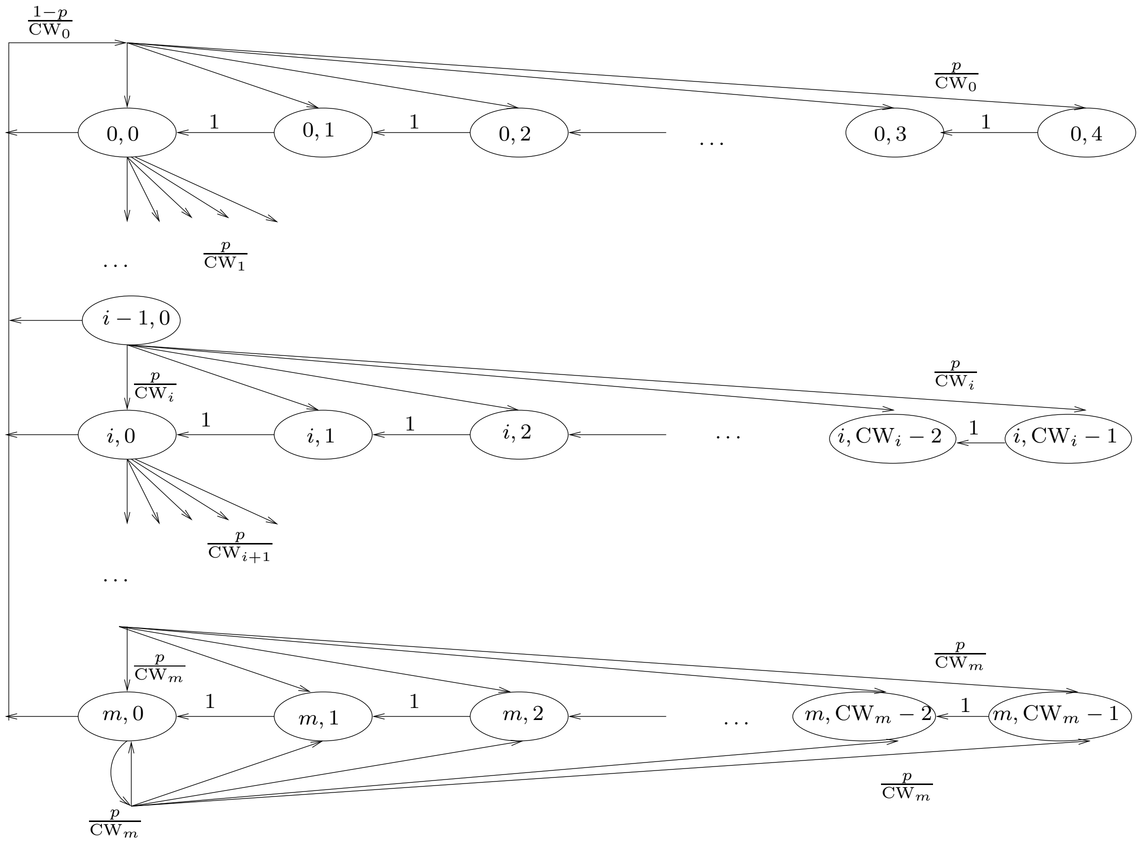


Figure 6.4: Bianchi's model: a Markov chain to model the backoff window

$$b_{0,0} = \frac{2(1-2p)(1-p)}{(1-2p)(CW_{\min}+1) + pCW_{\min}(1-(2p)^m)}. \tag{6.1}$$

The probability τ that the station transmits a packet in a generic slot time is then derived as the sum $\tau = \sum_{i=0}^{m} b_{i,0}$, as any tranmission occurs when the backoff is zero. From $b_{i,0} = p^i b_{0,0}$ then follows, as derived in [14, Eqn. (7)]:

$$\tau = \sum_{i=0}^{m} b_{i,0} = \frac{b_{0,0}}{1-p} = \frac{2(1-2p)}{(1-2p)(CW_{\min}+1) + pCW_{\min}(1-(2p)^m)}. \tag{6.2}$$

Due to the assumption of n identical, independent stations, the conditional collision probability p can be rewritten as the probability that at least one of the remaining $n-1$ stations transmits, as shown in [14, Eqn. (9)]:

$$p = 1 - (1-\tau)^{n-1}. \tag{6.3}$$

These two mutually dependent stationary probabilities τ and p are then expressed in the form of a system of two non-linear equations. Bianchi proves that this system has a unique solution [14].

Recall, that in a generic slot time three different events can occur: the successful transmission of a packet, a collision, or just an empty slot used for counting down backoff. Note that the length of the generic slot time depends on the event that occurs. Considering the different durations and probabilities of these events, it is possible to compute the throughput of the system, as will be explained in Section 6.4.3.

6.4.2 The extended model by Engelstad et al.

In a recent paper [29], Engelstad et al. extend the model of Bianchi by including the impact of the QoS enhancements on the effectively available capacity in IEEE 802.11e. Furthermore, this extended model allows to compute the throughput for stations from N different access categories, with n_i stations per category. The equations presented in [29] also hold in the non-saturated case, i.e., when the stations not necessarily always have a next packet to send.

In our decomposition analysis approach, however, we still use the saturated case. Due to our parameter choices, an active source sends, on average, 500 packets in a row before becoming inactive; this means that, on average, with probability $\frac{499}{500}$ there is a next packet to be sent. Due to the decomposition, inactive sources are not considered in the packet-level model, but they are accounted for at flow level. This might be seen as an approximation, but its impact will be small, as we will see later.

For our purposes, we use two different $\underline{a}$ccess $\underline{c}$ategories, AC$_i$ with $i = b$ for the bottleneck, and $i = s$ for the active sources, where AC$_b$ contains zero or one station, and AC$_s$ contains zero to ten stations. Note that access categories are not assigned on a per flow basis, but on a per node basis. So, each node has only one access category, however, packets change their access category when they arrive at the bottleneck.

Engelstad et al. now proceed to compute similar probabilities as Bianchi does, however, now for each possible access category. Simplified for the saturated case, the steady-state probability for category i with zero backoff in the first backoff stage $b_{i,0,0}$ is derived as follows in [29, Eqn. (10)]:

$$\frac{1}{b_{i,0,0}} = \sum_{j=0}^{m} \left(1 + \frac{1}{1 - p_i^*} \sum_{k=0}^{CW_{i,j}-1} \frac{CW_{i,j} - k}{CW_{i,j}} \right) p_i^j, \tag{6.4}$$

where p_i^* is just p_i. However, if AIFS differentiation is used, p_i^* is defined differently, as explained below in Equation 6.12.

Then τ_i, the probability that a station of category i transmits, and p_i, the probability that a transmission of category i is successful, are computed. As in the Bianchi model, these probabilities are defined through a system of $2N$ mutually dependent non-linear Equations [29, Eqn. (5) and Eqn. (12)], which can easily be solved using a tool like Maple. In the following, we repeat these equations, however, simplified for the saturated case and with zero dropping probability:

$$\tau_i = \frac{b_{i,o,o}}{1-p_i} \text{ and } p_i = 1 - \frac{1-p_b}{1-\tau_i}, \tag{6.5}$$

where $p_b = 1 - \prod_{i=0}^{N-1}(1-\tau_i)^{n_i}$ denotes the probability that the channel is busy. Note that we have to solve these equations once for every combination of number of active nodes in each access category, that is, AC$_b$ and AC$_s$, to obtain throughputs for every possible combination of active nodes in the high-level model.

From the above probabilities τ_i, p_b and p_i, Engelstad et al. then derive yet another two probabilities. The probability $p_{i,s}$ for a successful transmission for category i is computed as follows in [29, Eqn. (27)]:

$$p_{i,s} = \frac{n_i \tau_i}{(1-\tau_i)} \prod_{c=0}^{N-1} (1-\tau_c)^{n_c}, \tag{6.6}$$

and the probability p_s for successful transmission[1] for any category is derived in [29, Eqn. (28)] as :

$$p_s = \sum_{i=0}^{N-1} \frac{n_i \tau_i}{(1-\tau_i)} \prod_{h=0}^{N-1} (1-\tau_h)^{n_h}. \tag{6.7}$$

6.4.3 Relative throughputs

Using the probabilities $p_{i,s}$, p_s, and p_b obtained from the model of [29], we can derive the actual throughput for each access category, i.e, the throughput of the bottleneck, and the throughput of all active stations, as follows:

$$S_i = \frac{p_{i,s} \cdot \mathrm{txop}_i \cdot t_{\mathrm{data}}}{(1-p_b)\cdot t_{\mathrm{slot}} + p_s \cdot T_s + (p_b - p_s)\cdot T_c}, \; i \in \{b, s\}, \tag{6.8}$$

where the denominator states the average duration of a generic time slot, being the sum of the times for an empty slot (t_{slot}), the time for a successful transmission (T_s), and the time for a collision (T_c), weighted by their respective probability. The nominator denotes the part of a time slot that is, on average, used for transmission of data for category i. Here, txop$_i$ denotes the number of packets of length t_{data}

[1]Note that the probability p_s as presented by Engelstad et al. in [29, Eqn. (28)], does not directly correspond to the probability P_s as presented by Bianchi in [14, Eqn. (11)], as the latter is conditioned on the fact that at least one station transmits.

sent after winning contention. Instead of modeling TXOP$_{\text{limit}}$ as a time period, we assume that for a given packet size, a station of category i can send txop$_i$ packets during one TXOP period. That is, we model a time-based mechanism by means of a count-based mechanism; a similar modeling approach has been applied successfully, for instance, by Groenendijk [35], to model time-token access mechanisms. Note that (6.8) above provides the input for the state-dependent transition rates in the iSPN model, cf. Table 6.2.

To actually compute the values, Table 6.3 specifies the individual durations and the default values for the system parameters that are used to compute the time for a successful transmission and the time for a collision, respectively. Considering the use of <u>R</u>equest <u>T</u>o <u>S</u>end/<u>C</u>lear <u>T</u>o <u>S</u>end (RTS/CTS), the time for a collision is given by

$$T_c = t_{\text{PHY}} + t_{\text{RTS}} + t_{\text{AIFSmin}}. \tag{6.9}$$

The time for the successful transmission of a packet with RTS/CTS depends on the TXOP values:

$$\begin{aligned} T_s = t_{\text{PHY}} + t_{\text{RTS}} + t_{\text{SIFS}} + t_{\text{PHY}} + t_{\text{CTS}} + \\ (t_{\text{SIFS}} + t_{\text{PHY}} + t_{\text{MAC}} + t_{\text{data}} + t_{\text{SIFS}} + t_{\text{PHY}} + t_{\text{ACK}}) \cdot \overline{\text{txop}} + t_{\text{AIFSmin}}, \end{aligned} \tag{6.10}$$

where $\overline{\text{txop}}$ is the average number of packets sent after a successful contention computed as weighted sum:

$$\overline{\text{txop}} = \sum_{i \in \{s,b\}} \text{txop}_i \cdot \frac{p_{i,s}}{p_s}. \tag{6.11}$$

parameter	value	comments
t_{SIFS}	$10\mu s$	
t_{slot}	$20\mu s$	
t_{PHY}	$192\mu s$	assuming long preamble
t_{RTS}	$160\mu s$	20 bytes @ 1 Mbps
t_{CTS}	$112\mu s$	14 bytes @ 1 Mbps
t_{MAC}	$25\mu s$	34 bytes @ 11 Mbps
t_{data}	$1091\mu s$	1500 bytes @ 11 Mbps
t_{ACK}	$112\mu s$	14 bytes @ 1 Mbps
t_{AIFSmin}	$50\mu s$	$2 \cdot t_{\text{slot}} + t_{\text{SIFS}}$

Table 6.3: Time durations & default values

The relative share of capacity for the bottleneck (S_b) and for the sources (S_s), as used in Table 6.3 is given by the throughput per access category as stated in Equation (6.8) for the different number of stations.

The different values of CW$_{\min}$, CW$_{\max}$ are immediately taken into account, when computing τ_i and p_i, as described in [29].

Modeling differentiation by means of $\text{TXOP}_{\text{limit}}$ is also incorporated in Equation (6.8), which is an extension of the throughput as presented in [29].

Table 6.4 specifies the values for the differentiation parameters for the bottleneck and the sources that are considered in the following. The default values are marked with an asterisk.

parameter	value for AC_b	value for AC_s
AIFS_b (slots)	2*	2*, 7 or 12
CW_{min} (slots)	32*	32*, 64 or 128
$r_{\max}$	4*	4*
txop (packets)	1*, 2 or 3	1*

Table 6.4: Considered values for the differentiation parameters per access category

AIFS approximation

When modeling access categories with different AIFS, we use the approximation as proposed in [29, Section 3.3] to compute the throughput.

The differentiation of the AIFS is taken into account when computing p_b and p_i, as follows. Engelstad et. al. distinguish between the following two probabilities: A generic slot is sensed busy with probability p_i upon transmission and with probability p_i^* during backoff. AIFS differentiation is then modeled by adjusting the countdown blocking probability p_i^*.

Without AIFS differentiation p_i simply equals p_i^*. When AIFS differentiation is used, p_i^* is set to p_i for the category with the smallest arbitration interframe spacing, denoted AIFS_{min}. For all other categories the AIFS value is reduced by AIFS_{min}:

$$A_i = \text{AIFS}_i - \text{AIFS}_{\text{min}}.$$

The remaining slots A_i that stations of category i have to wait longer before backoff countdown, are modeled as distributed randomly and distributed uniformly over all slots. Thus, the probability that a channel is busy, p_b, is replaced by the new scaled expression $(\text{A}_i + 1)p_b$. Together with a minimum bound on the probability, the following expression for p_i^* is obtained in [29, Eqn. (15)]:

$$p_i^* = \min\left(1, p_i + \frac{A_i p_b}{1 - \tau_i}\right), \tag{6.12}$$

which has to be used in (6.4). This implies that the AIFS_{min} is used for computing the time for a successful transmission (cf. (6.10)) and the time for a collision (cf. (6.9)).

6.4.4 Measures of interest

Instead of specifying the performance models of interest manually at the state level, we use a high-level specification mechanism. We specify the iSPN and the relative throughputs in CSPL that is a C-based specification mechanism for stochastic Petri nets. Using the CSPL implementation [12], we obtain the underlying infinite CTMC. The resulting QBD is given in Figure 6.5. Every level consists of $K + 1$ states, modeling the number of active sources. Whenever at least one source is present, packets can arrive and whenever at least one packet is present, this packet can be served.

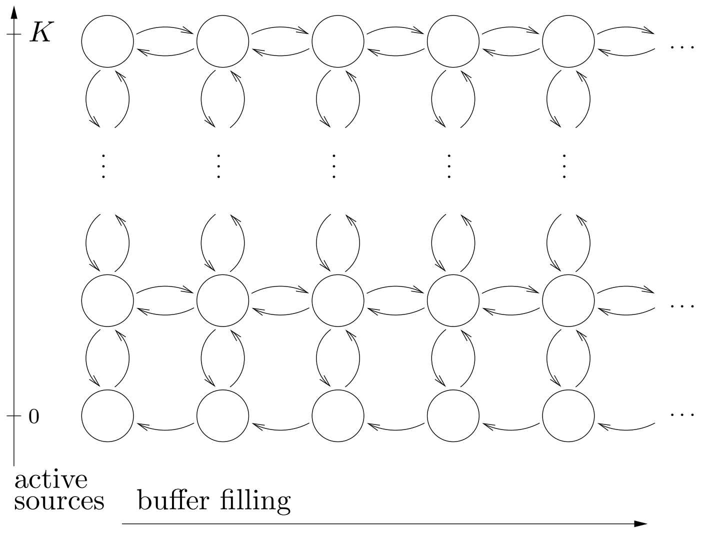


Figure 6.5: Underlying QBD of the bottleneck model

The bridge B is the bottleneck of the two-hop ad hoc network. We therefore study the expected number of packets in the buffer of the bottleneck (M1), as well as the throughput of the bottleneck (M2), and the expected number of active sources (M3), for all possible QoS mechanisms and for different source arrival rates.

All three measures can be expressed using the CSRL expected reward operator (c.f. Section 5.3), $\mathcal{E}_{\leq r}(\Phi)$, where Φ is just *true*. Recall, that the expected reward operator combines the steady-state probability to be in a state s with the reward $\rho(\mathsf{s})$.

(M1) We choose the reward to be the number of packets currently in the buffer, which is just the level-index in the QBD, so that, we have a level dependent reward.

(M2) The value of the transition *output* is chosen as reward. This value depends on the number of active sources and on whether or not at least one packet is currently in the buffer. In the model, the number of active sources is the same in corresponding states of different levels, so that we have level-independent rewards in this case.

(M3) We assign to each state the number of sources that is currently active as reward. This, again results in a level independent reward.

Note, that we are not able to compute flow-related measures, as the flow time or flow throughput, as our model does not distinguish between packets of the different sources. However, we think that the throughput at the bottleneck and, possibly, the throughput at the sources is enough to capture the effects of differentiation in this scenario.

6.5 Simulation model

The bottleneck scenario has also been modeled and simulated in Opnet, version 11.5. [62]. In this section we explain the simulation setup and the parameter settings for the different QoS parameters.

Opnet is organized hierarchically in levels, where every level adds more detail. At the highest level, we place ten *advanced WLAN stations*, as provided by Opnet, to model the ten sources, and two more to model the bottleneck and the sink, where all the data is sent to. We then adapt the WLAN stations in the node editor once for the sources and once for the bottleneck. For the sources, a new traffic generation process is created that is put inside each source. The traffic generation process can be either active or inactive. As soon as it becomes active it places a geometrically distributed amount of packets into its MAC layer queue. When the transmitter has been able to send all these packets to the bottleneck it gets inactive again. The packet size is set to 1500 bytes. As in the analytical model all sources are independent and become active with the global arrival rate λ, when currently inactive.

The WLAN station that models the bottleneck does not need a traffic generation process. Arriving packets from the MAC layer are immediately routed back to the MAC layer and are forwarded to the sink. The size of the data buffer in the MAC layer is set to the highest possible value, 10^8, to match the assumption of the infinite buffer in the analytical model as accurately as possible. Once the buffer limit is reached, data packets will be discarded until the buffer has free space to store new packets. Note that the complete access mechanism, in all its details, is included in the Opnet simulation. No approximations or further assumptions are being done. The *wireless LAN parameters* in Opnet are set as follows:

- the data rate is set to 11 Mbps,
- regular RTS/CTS is enabled, the RTS threshold is set to 256 bytes,

- EDCA parameters are not supported for basic IEEE 802.11 and supported for IEEE 802.11e and set according to the simulation scenario under study.

The content of the MAC layer queue is sampled over time and its time average corresponds to the measure *expected buffer occupancy* (M1) in the analytical model. The number of currently active sources is also sampled and its time average corresponds to *expected number of active sources* (M3) in the analytical model. Furthermore, the throughput of the bottleneck is sampled over time and its time average is compared to the *throughput* (M3) of the bottleneck in the analytical model. For every QoS parameter we consider seven different loads:

$$\lambda \in \{0.01; 0.015; 0.02; 0.025; 0.03; 0.035; 0.04\}.$$

Every value of λ is simulated in every scenario with ten randomly chosen seeds for two hours, leading to 70 simulation runs per curve. One simulation run takes between 20 and 50 minutes, resulting in an estimated run time per point, including confidence intervals, of 200 to 500 minutes. In the following we show the mean of the simulation results for ten seeds together with the corresponding 95% confidence interval.

6.6 Comparing analytical and simulation results

We compare the analytical results with the simulation results for the basic scenario in Section 6.6.1, when differentiating with CW_{min} in Section 6.6.2, when differentiating with AIFS in Section 6.6.3 and finally for differentiating with TXOP in Section 6.6.4. We compare the analytical results with the simulation results for the throughput of the bottleneck that is obtained with four different parameter settings in Section 6.6.5.

6.6.1 Basic scenario

In the basic 802.11 scenario the EDCA parameters are set to the default values, as given in Table 6.4. Figure 6.6 shows the expected buffer occupancy in IEEE 802.11 without differentiation and Figure 6.7 shows the expected number of active sources. In the basic scenario the results from simulation and analysis are very close to each other, the estimated expected buffer occupancy is always inside the confidence intervals. With increasing λ the variance of the simulated buffer occupancy grows as seen from the larger confidence intervals.

As could be expected, the buffer occupancy increases exponentially with λ. For λ larger than 0.04 the system soon becomes overloaded. The steady-state distribution then does not exist any more and the buffer of the bottleneck in the simulation overflows. The number of active sources grows almost linearly with increasing λ.

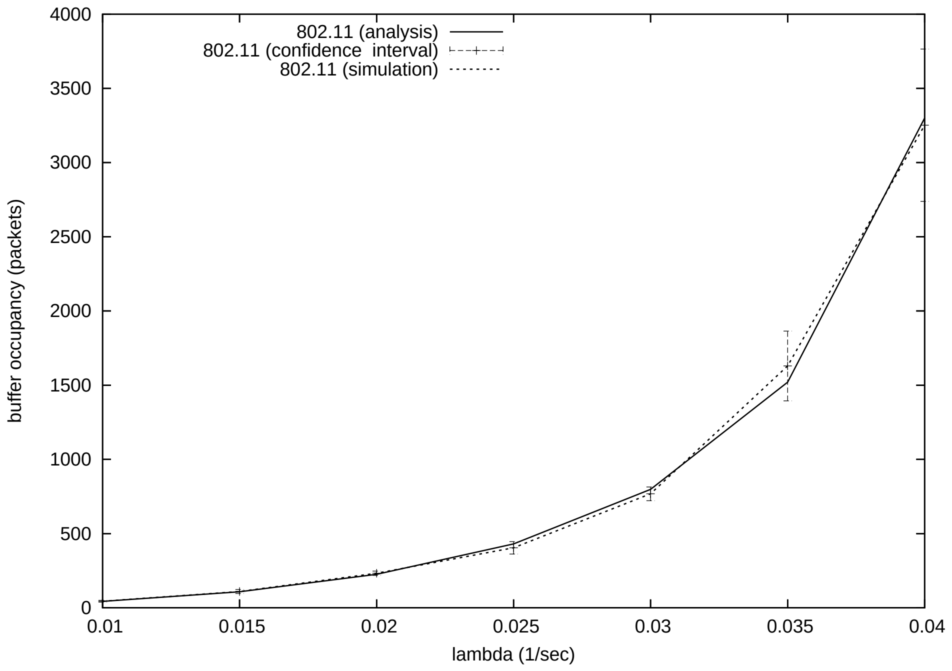


Figure 6.6: Expected buffer occupancy in the bottleneck with IEEE 802.11.

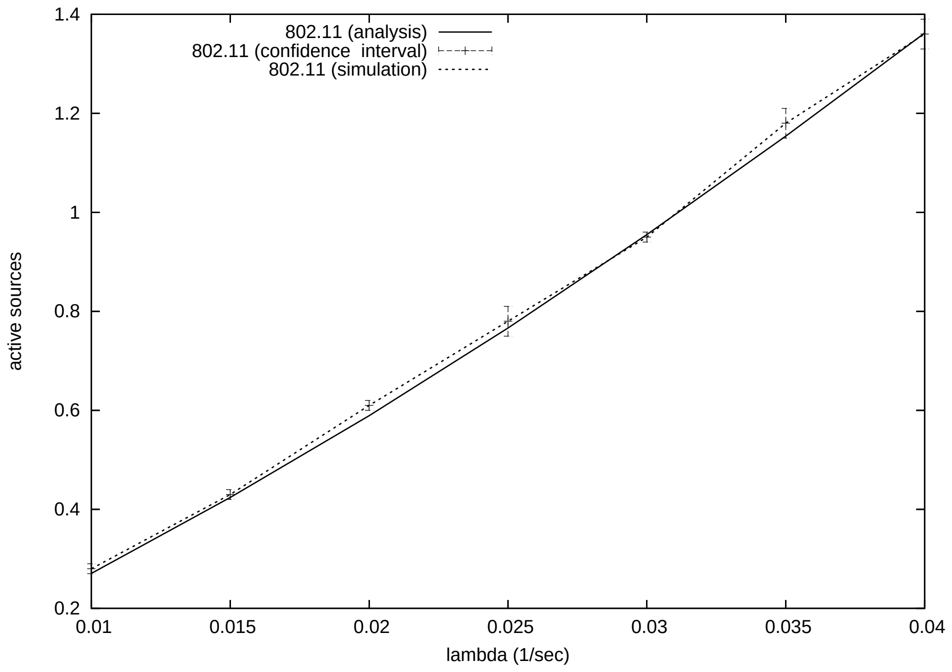


Figure 6.7: Number of active sources with IEEE 802.11.

6.6.2 Differentiating with $\mathrm{CW}_{\mathrm{min}}$

We compare the results from simulation and analysis for two different settings for the expected buffer occupancy in Figure 6.8 and in Figure 6.9 for the expected number of active sources. In this scenario, the contention window of the sources $\mathrm{CW}_{\mathrm{min},s}$ is set to 64 and to 128, whereas the other EDCA parameters are set to the default values as specified in Table 6.4. Compared to basic IEEE 802.11 the mean buffer occupancy is lower when the sources operate with a larger window size. This is due to the fact that the bottleneck gets a higher capacity share. On the other hand the sources remain active longer (expected number of sources is higher than in the basic scenario).

We observe that in the parameter setting $\mathrm{CW}_{\mathrm{min},s} = 64$, simulation and analysis results are close for buffer occupancy and number of active sources, respectively. For the parameter setting $\mathrm{CW}_{\mathrm{min},s} = 128$, the analysis overestimates the capacity that is allocated to the bottleneck. Hence, the mean buffer occupancy is underestimated and the mean number of active sources is overestimated by the analysis.

6.6.3 Differentiating with AIFS

In this scenario the value of AIFS is changed first to 7 and then to 12 in the sources, while the other EDCA parameters remain set as in the basic scenario. The analytical results and the simulation results for the two different AIFS settings are compared in Figure 6.11 for the expected number of active sources. Simulation shows that our results are highly accurate.

Figure 6.10 shows the expected buffer occupancy at the bottleneck on a logarithmic scale. The mean buffer occupancy is much lower than in the basic scenario. It is also lower than in the CW scenario, while the mean number of active source is comparable. This indicates that in the CW scenario more capacity is wasted due to longer backoff times.

6.6.4 Differentiating with TXOP

To study the influence of TXOP_b this value is set to 4000 μs, and to 6000 μs in the bottleneck only, while the other EDCA parameters remain unchanged. A TXOP_b of 4000 microseconds allows two data packets to be sent, whereas a TXOP_b of 6000 microseconds allows three data packets to be sent. The resulting curves from analysis and simulation are compared in Figure 6.12 for the buffer occupancy and in Figure 6.13 for the number of active sources, respectively. The mean buffer occupancy is much lower than in all other scenarios. The number of active sources is lower than in the other differentiated scenarios and only slightly higher than in the basic scenario. This is due to the fact that less packets have to undergo contention and thus less collisions occur with leads to a higher effective capacity.

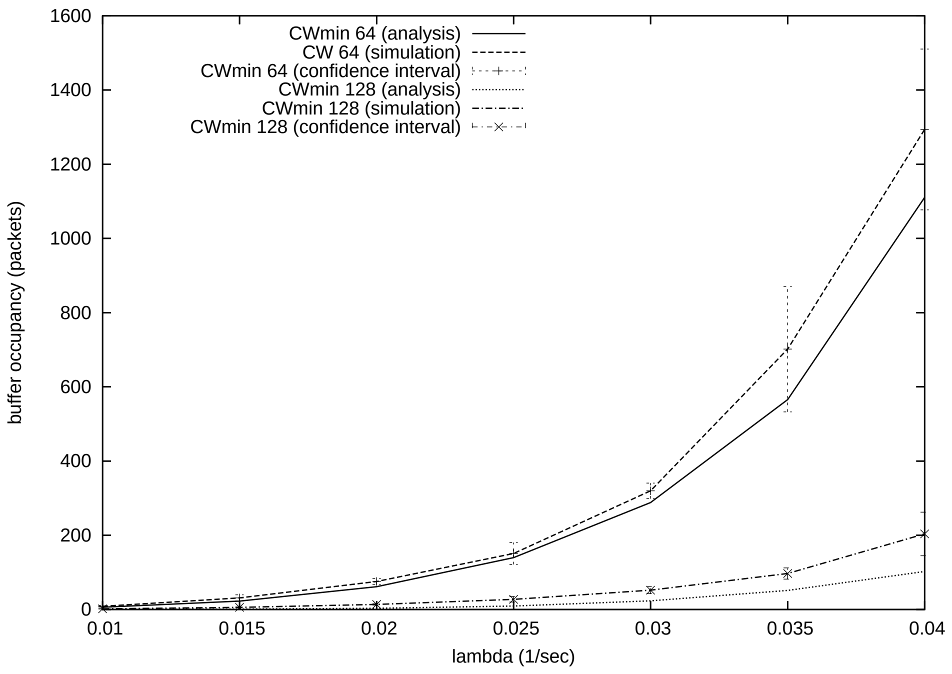


Figure 6.8: Expected buffer occupancy for different $CW_{\min,s}$ and $CW_{\min,b} = 32$.

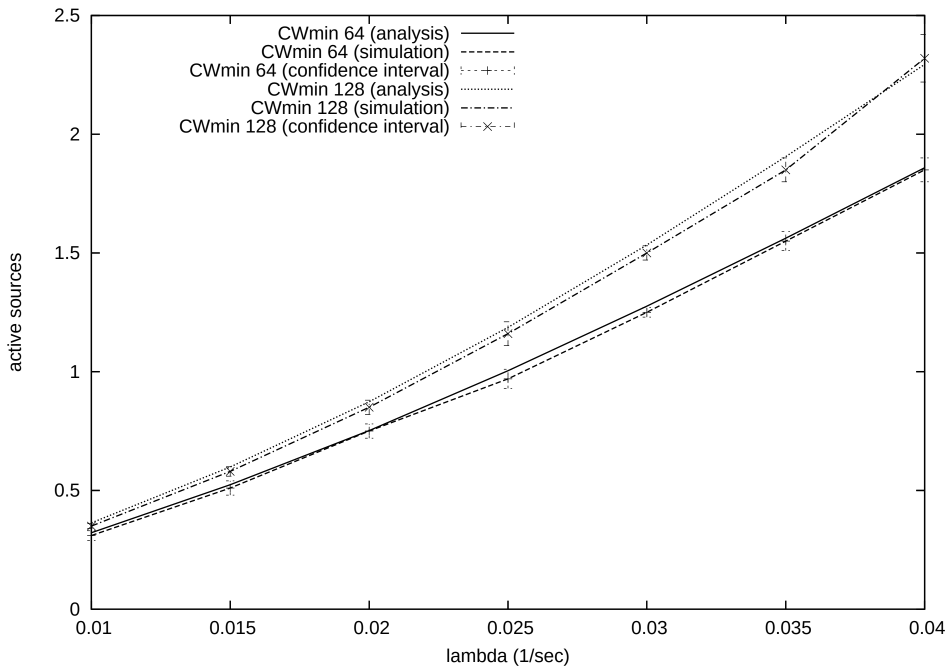


Figure 6.9: Expected number of active sources for different $CW_{\min,s}$ and $CW_{\min,b} = 32$.

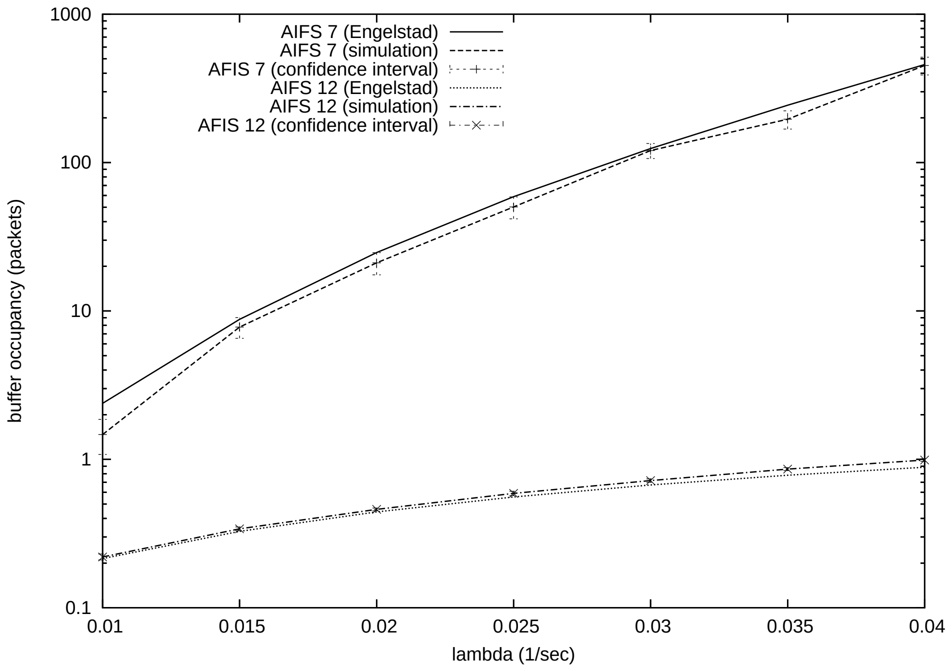


Figure 6.10: Expected buffer occupancy for different AIFS$_s$ and AIFS$_b = 2$.

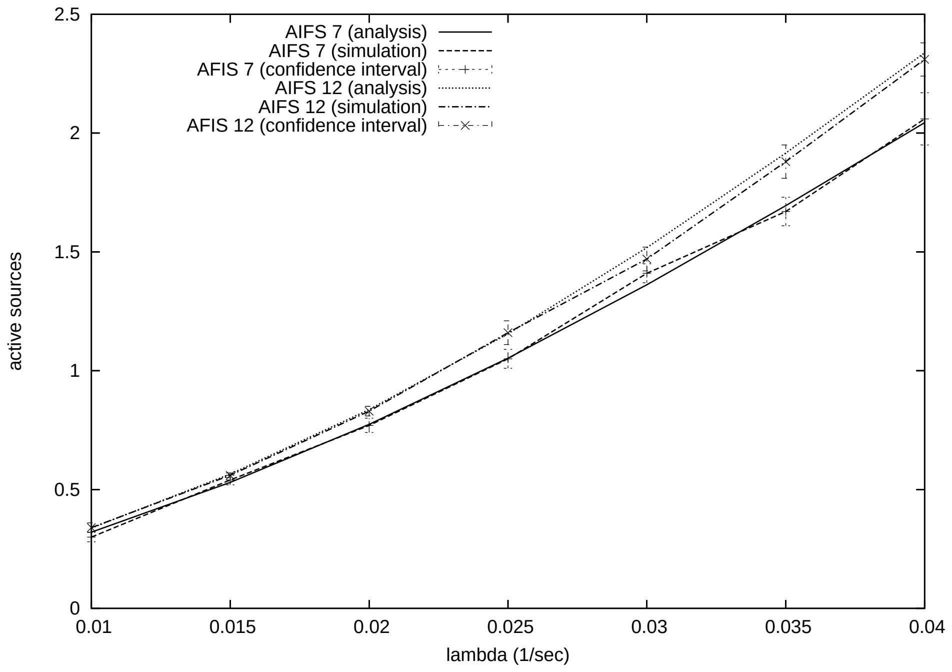


Figure 6.11: Expected number of active sources for different AIFS$_s$ and AIFS$_b = 2$.

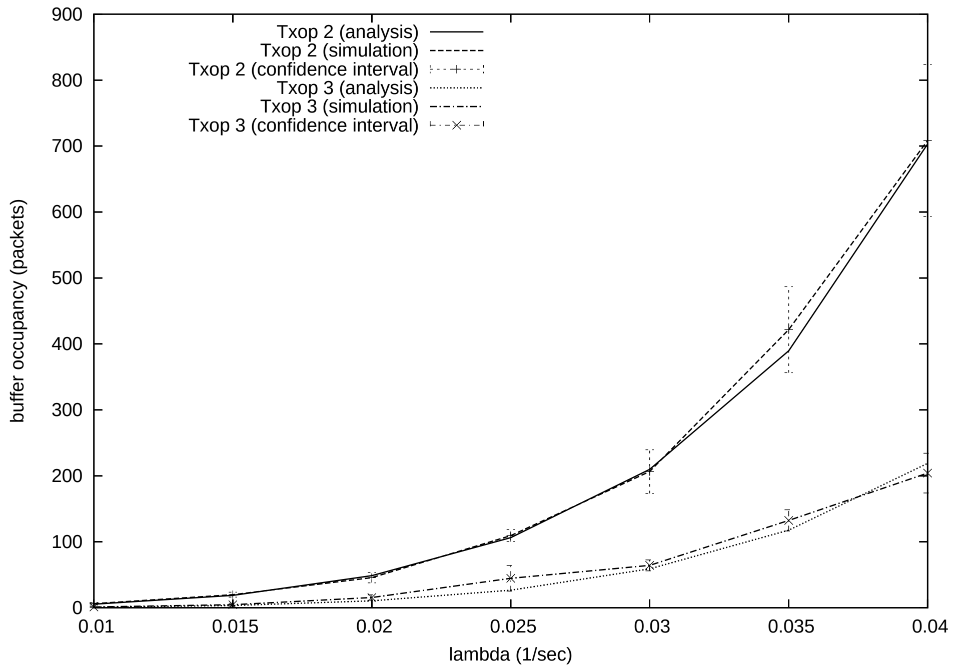


Figure 6.12: Expected buffer occupancy for different TXOP_b and $\mathrm{TXOP}_s = 1$.

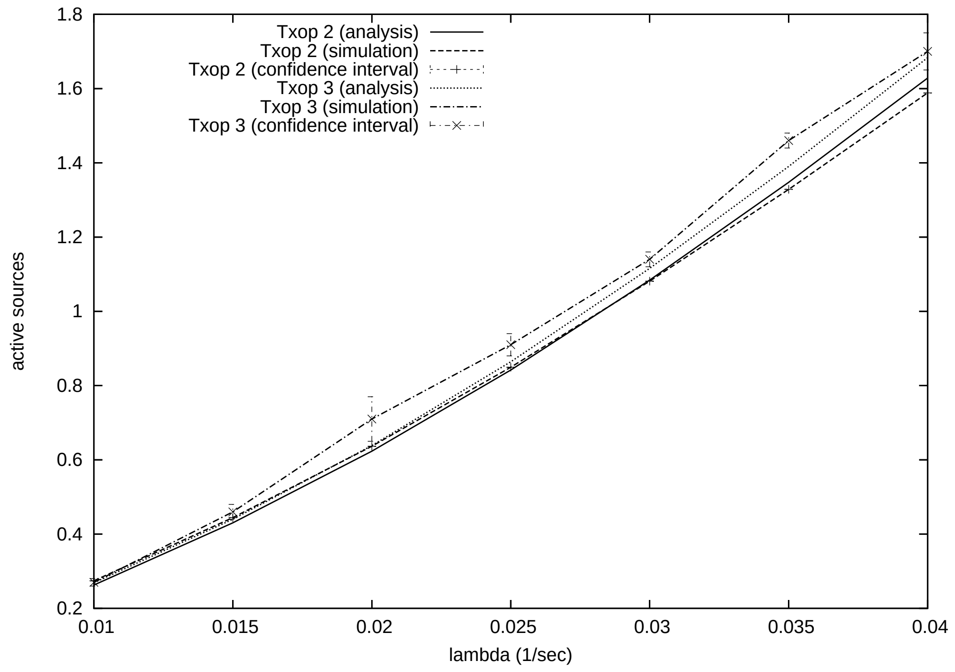


Figure 6.13: Expected number of active sources for different TXOP_b and $\mathrm{TXOP}_s = 1$.

6.6.5 Throughput of the bottleneck

To further analyze the impact of the differentiation parameters, we compute the throughput of the bottleneck, as a function of the parameter λ. The throughput for basic IEEE 802.11 is compared with the throughput that is achieved in three differentiated settings.

In Figure 6.14 we show our highly accurate analytical results for the throughput in these different settings together with the confidence intervals of the corresponding simulation runs for three different settings: where TXOP_b is set to 3, when $\mathrm{CW}_{\mathrm{min},s}$ is set to 128, when AIFS_s is set to 12 and for basic IEEE 802.11.

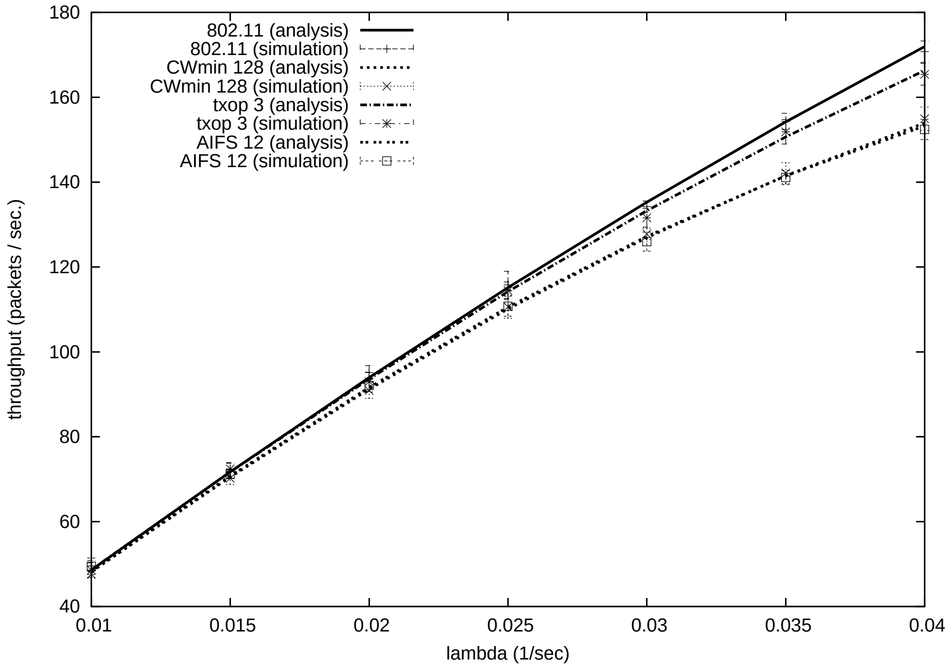


Figure 6.14: Throughput of the bottleneck

For all $\lambda \in \{0.01, \ldots, 0.04\}$, the largest throughput is achieved in basic IEEE 802.11. The throughput for $\mathrm{TXOP}_b = 3$ is just a little lower for increasing values of λ. The throughputs for differentiated $\mathrm{CW}_{\mathrm{min},s}$ and AIFS_s fall together and are considerably lower than in the two other cases, for $\lambda > 0.025$.

This non-intuitive result is due to the fact that, for given λ, the offered load depends on the average number of active sources. Recall that the number of active sources is higher in the differentiated settings than in basic IEEE 802.11, as part of the capacity has been moved from the sources to the bottleneck. When the sources remain active for a longer time, they become active less frequently, so in total fewer packets are sent to the bottleneck. Clearly, the bottleneck can only forward packets that have been sent to it from the sources. Since we assume infinite buffering capacity, the throughput equals the offered load as long as the bottleneck queue is stable. Note that with differentiation the queue remains stable for much higher λ, compared to the standard 802.11. As a result, the *maximum* throughput, for a given constraint on the buffer occupancy, using differentiation is expected to be higher than for the standard 802.11.

Finally, when $\mathrm{CW}_{\mathrm{min}}$ and AIFS are increased to differentiate, this comes at the cost of a decreased effective capacity, as more time slots will pass unused. In contrast, increasing TXOP effectively increases capacity, as multiple packets are sent within TXOP after just one contention period. However, as can be seen in Figure 6.14 the increase in effective capacity is not enough to compensate for the slightly higher mean number of active sources for different $\mathrm{TXOP}_{\mathrm{b}}$, as shown in Figure 6.13.

In the following, we analyze the maximum throughput that can be achieved per differentiation parameter for a given threshold on the average buffer occupancy of the bottleneck, for arbitrary values of λ.

6.7 Setting the parameters right

In this section we compute the maximum throughput that can be achieved for a given constraint on the buffer occupancy, per differentiation parameter. Note that we only differentiate one parameter at a time. For different AIFS, this is discussed in Section 6.7.1, in Section 6.7.2 for different $\mathrm{CW}_{\mathrm{min}}$ and in Section 6.7.3 for different TXOP. Finally we compare the maximum throughput that can be obtained with the different QoS parameters with the maximum throughput obtainable with basic IEEE 802.11 in Section 6.7.4. The numerical values and the graphical representations in this section have been produced by Jesper Bax as part of his ongoing bachelor thesis.

6.7.1 Throughput for different AIFS

The constrained maximum throughput of a given combination of AIFS_b and AIFS_s is obtained as illustrated in Figure 6.15. First the value of λ is identified for which the buffer occupancy equals the given threshold. In Figure 6.15 the buffer occupancy for $\lambda = 0.024$ equals the threshold of 50 packets. Then the corresponding throughput of 108 packets per second for this value of λ is computed.

Figure 6.16 shows the maximum throughput that can be achieved per parameter setting when the buffer occupancy is bound to be at most 50. AIFS_b is differentiated between 2 and 9 and AIFS_s is differentiated between 5 and 12.

For combinations of large AIFS_b and small AIFS_s the bound on the average buffer occupancy can only be met for $\lambda = 0$. Clearly, the resulting throughput is zero as well. For increasing values of AIFS_s the achievable throughput grows. The maximum throughput of 195.21 packets per second is achieved for $\mathrm{AIFS}_b = 2$ and $\mathrm{AIFS}_s = 10$, as marked with `x` in Figure 6.15. If AIFS_s is increased above 10 and AIFS_b above 2,

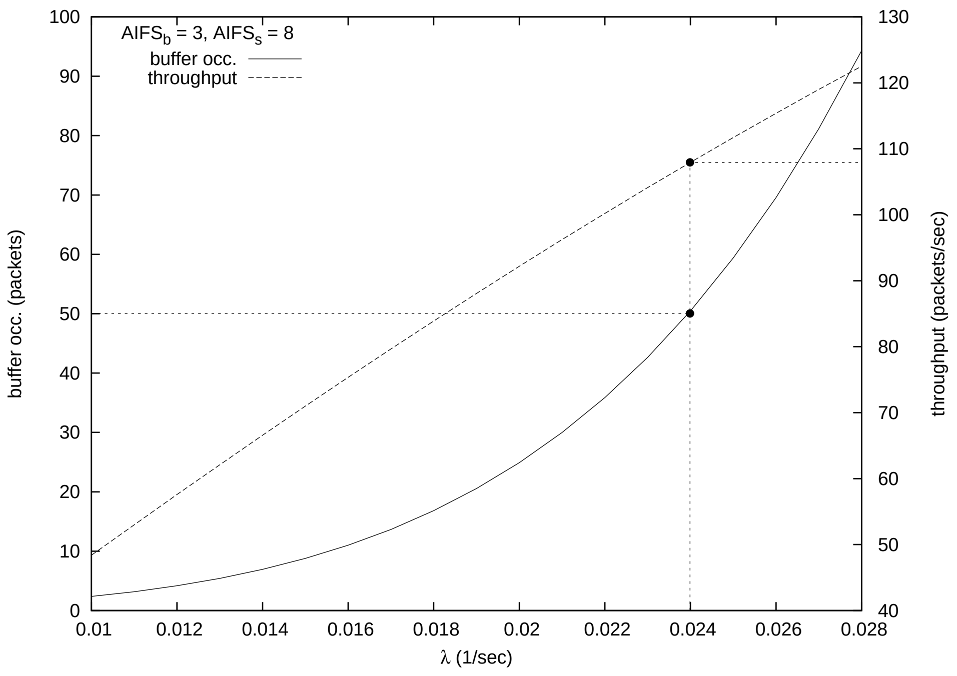


Figure 6.15: Average buffer occupancy versus throughput for a given parameter setting

the achieved throughput declines. This is due to the waste of capacity, as stations have to wait longer before they can start decrementing their backoff.

Figure 6.17 shows the maximum throughput that can be achieved for AIFS$_b = 2$, when AIFS$_s$ ranges between 2 and 12 slots and the threshold on the average buffer occupancy ranges between 10 and 100.

Again, the maximum throughput is 195 packets per second. This throughput is obtained for AIFS$_s = 10$ and AIFS$_b = 2$, independent of the bound on the throughput. When AIFS$_s$ is increased beyond the value ten, the throughput decreases due to the waste of capacity, evenly for all considered thresholds. When AIFS$_s$ is set smaller than ten, the throughput decreases overall and even faster for smaller thresholds. Only small values of λ meet the low threshold on the average buffer occupancy. However, this keeps the throughput low, as well. For several combinations of small AIFS$_s$ and low thresholds the value of λ even has to be zero to match the constraint on the buffer occupancy, resulting in zero throughput.

Concluding, we can state that a maximum throughput of 195 packets per second can be achieved, when differentiating AIFS. Moreover, this maximum is independent of the threshold on the average buffer occupancy. Regarding the throughput and the buffer occupancy, AIFS$_s$ should be chosen rather too big than too small, while AIFS$_b$ should be set to 2.

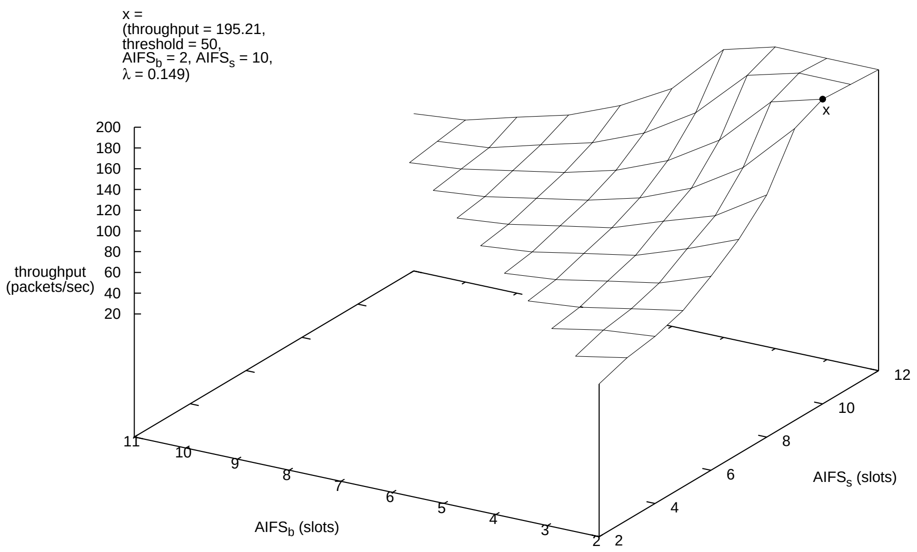


Figure 6.16: Maximum throughput for different combinations of AIFS$_b$ and AIFS$_s$

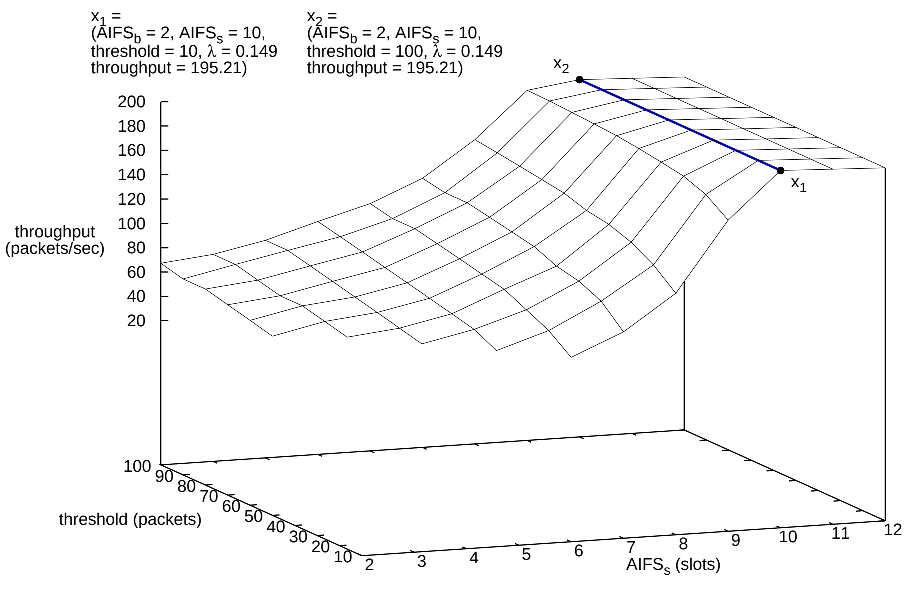


Figure 6.17: Maximum throughput for different combinations of AIFS$_s$ and the threshold on the average buffer occupancy

6.7.2 Throughput for different $\mathrm{CW}_{\mathrm{min}}$

Figure 6.18 shows the maximum throughput that can be achieved for different combinations of $\mathrm{CW}_{\mathrm{min},b}$ and $\mathrm{CW}_{\mathrm{min},s}$, when the average buffer occupancy is bound to 50. $\mathrm{CW}_{\mathrm{min},b}$ ranges between 31 and 287 and $\mathrm{CW}_{\mathrm{min},s}$ ranges between 31 and 447. The maximum throughput of 193 packets per second is obtained for $\mathrm{CW}_{\mathrm{min},b} = 31$ and $\mathrm{CW}_{\mathrm{min},s} = 255$ (point x in Figure 6.18). For higher values of $\mathrm{CW}_{\mathrm{min},b}$ the throughput decreases due to several reasons: first, capacity is wasted as randomly chosen backoffs become unnecessarily large, second the difference between $\mathrm{CW}_{\mathrm{min},b}$ and $\mathrm{CW}_{\mathrm{min},s}$ is too small, resulting in already high buffer occupancy for still small values of λ. Consequently the throughput remains small. For the same reason, several combinations of high $\mathrm{CW}_{\mathrm{min},b}$ and low $\mathrm{CW}_{\mathrm{min},s}$ result in zero throughput. When $\mathrm{CW}_{\mathrm{min},s}$ is increased above 255, the throughput decreases slowly, as capacity as wasted due to large backoffs in the sources.

Figure 6.19 shows the maximum throughput that can be achieved when $\mathrm{CW}_{\mathrm{min},s}$ ranges from 31 to 447 and the bound on the average buffer occupancy ranges from 10 to 100. The throughput increases evenly for larger values of $\mathrm{CW}_{\mathrm{min},s}$. The maximum throughput is obtained for $\mathrm{CW}_{\mathrm{min},b} = 31$ and $\mathrm{CW}_{\mathrm{min},s} = 255$ and a threshold on the buffer occupancy of at least 40 packets. For values of $\mathrm{CW}_{\mathrm{min},s}$ above 255 the throughput decreases slowly, due to the waste of capacity. We can conclude that maximum throughput is obtained for $\mathrm{CW}_{\mathrm{min},b} = 31$ and $\mathrm{CW}_{\mathrm{min},s} = 255$ and a threshold of at least 40 packets. As for AIFS_s, the parameter $\mathrm{CW}_{\mathrm{min},s}$ should be chosen rather too big than too small.

6.7.3 Throughput for different TXOP

Figure 6.20 shows the maximum throughput that can be obtained for different combinations of TXOP_b and TXOP_s, when the average buffer occupancy has to be at most 50 packets. When TXOP_b ranges between 1 and 30 and TXOP_s between 1 and 15 the maximum of 281.103 packets is reached for $\mathrm{TXOP}_b = 30$ and $\mathrm{TXOP}_s = 4$. The maximum throughput, obtained when differentiating TXOP is almost 50% higher than when differentiating AIFS or $\mathrm{CW}_{\mathrm{min}}$. On the one hand every increase in TXOP_b leads to an increase in the effective capacity. On the other hand the choice of TXOP_s highly depends on the value of TXOP_b, as can be seen in Figure 6.20. Again, combinations of small TXOP_b and large TXOP_s lead to zero throughput, because the constraint on the buffer occupancy cannot be met. Figure 6.21 shows the maximum throughput that is obtained for TXOP_b ranging from 1 to 30 and the threshold on the average buffer occupancy ranging from 10 to 100. The throughput increases evenly for larger values of TXOP_b and for larger thresholds, and the maximum throughput of 283 packets per second is achieved for the largest considered $\mathrm{TXOP}_b = 30$ and the largest considered threshold of 100 packets. This is due to the fact that every increase in TXOP_b leads to an increased capacity.

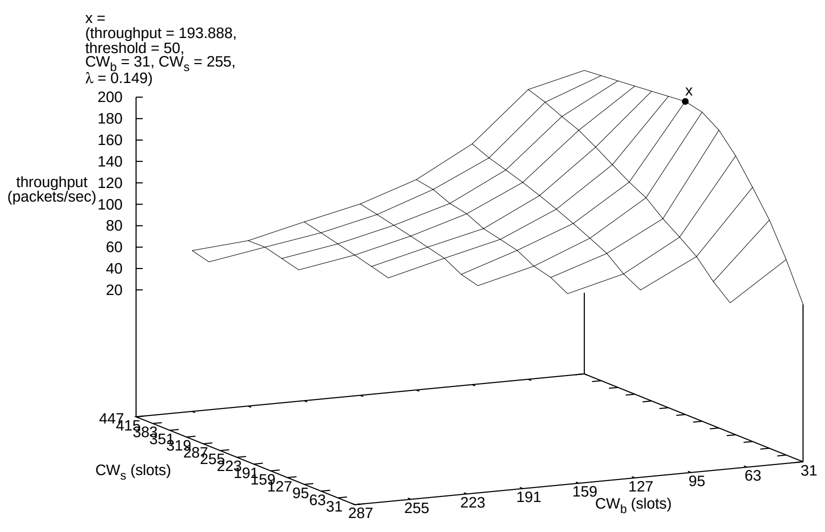


Figure 6.18: Maximum throughput for different combinations of CW$_b$ and CW$_s$

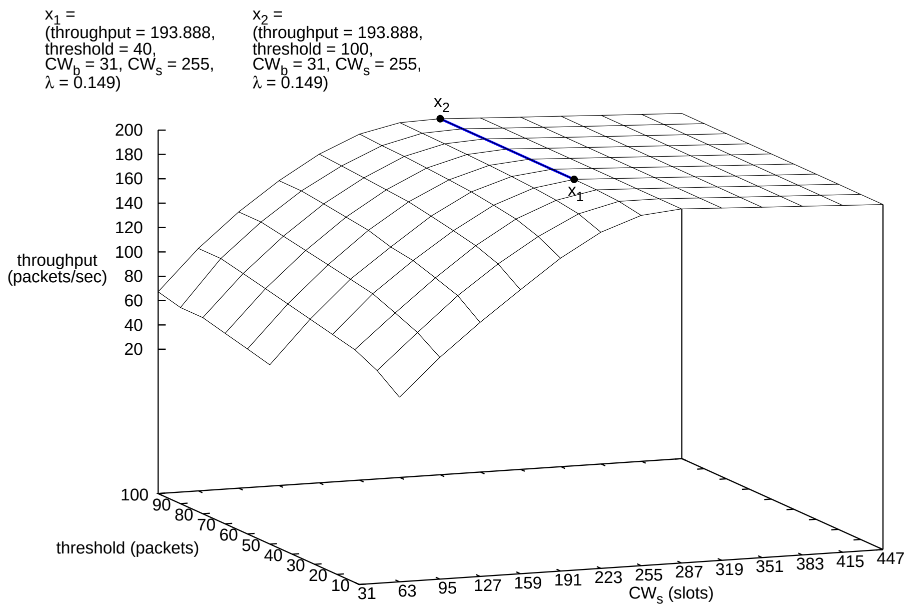


Figure 6.19: Maximum throughput for different combinations of CW$_s$ and the threshold on the average buffer occupancy

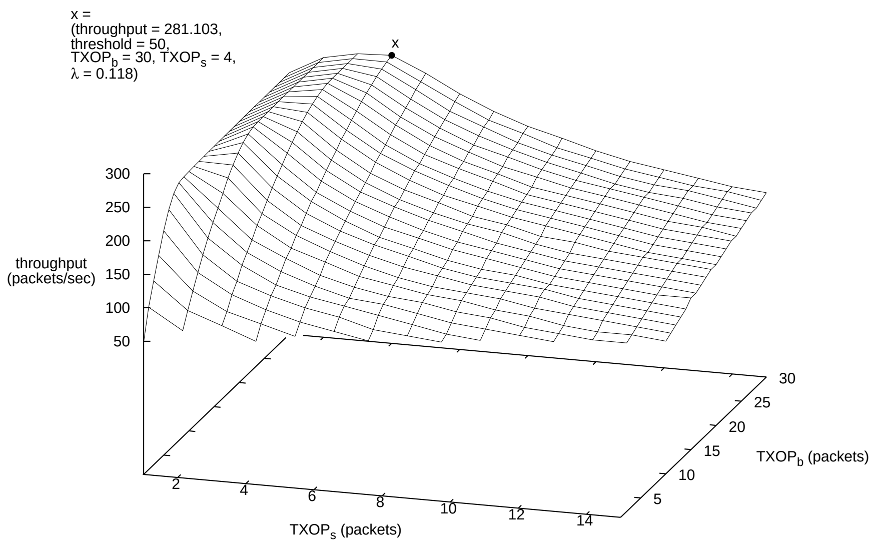


Figure 6.20: Maximum throughput for different combinations of TXOP$_b$ and TXOP$_s$

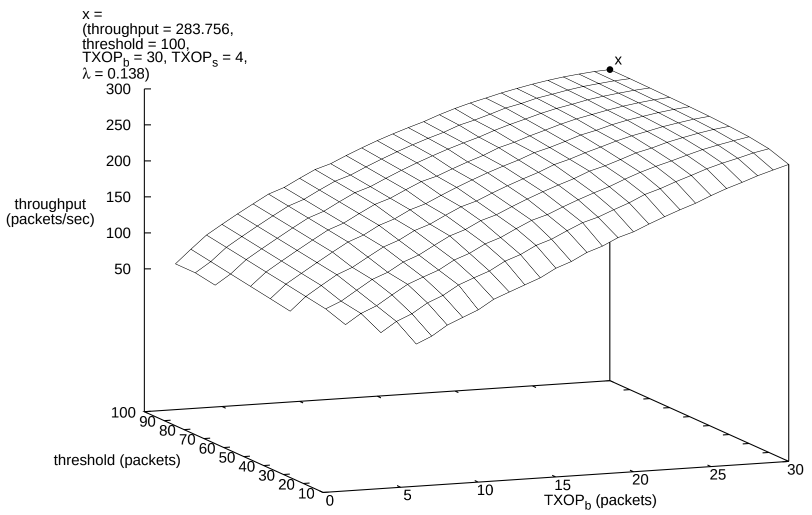


Figure 6.21: Maximum throughput for different combinations of TXOP$_b$ and the threshold on the average buffer occupancy

6.7.4 Overall comparison

To conclude this case study, we compare the maximum throughput that can be obtained for a given threshold on the buffer occupancy per differentiation parameter. Figure 6.22 shows this throughput as a function of λ under the constraint that the average buffer occupancy is smaller than 100 packets. All three differentiation parameters are able to keep the buffer occupancy below the given threshold of 100 packets for all considered values of λ. However, the throughput that can be obtained with differentiating TXOP$_b$ and TXOP$_s$ is about 50% higher for large values of λ than with differentiating AIFS or CW$_{\text{min}}$. Differentiating AIFS and CS$_{\text{min}}$ results in approximately the same maximum throughput. Note that the throughput obtainable with standard EDCA parameters is not included in this figure, as the buffer occupancy meets the constraint only for $\lambda < 0.015$.

Figure 6.23 shows the maximum throughput that can be obtained for the three differentiated settings and for basic IEEE 802.11 as a function of the threshold on the buffer occupancy. As one would expect, the smallest throughput is obtained in the non-differentiated setting. The throughput that can be obtained when differentiating AIFS and CW$_{\text{min}}$ is about the same. The highest throughput is obtained for differentiating TXOP$_b$ and TXOP$_s$. Note that we consider only thresholds between 45 and 100 packets, as smaller thresholds cannot be met with non-differentiated EDCA parameters. The throughput in the differentiated cases is almost independent of the allowed threshold, whereas the throughput in the basic setting grows slightly with growing thresholds on the buffer occupancy.

We state that in such a two-hop bottleneck scenario it is advisable to differentiate, using TXOP$_b$ and TXOP$_s$, as increasing these differentiation parameter results in an increase of the effective capacity. Differentiating TXOP$_b$ and TXOP$_s$ results in a maximum throughput that is 300% larger than the throughput in the non-differentiated setting and about 50% larger than when differentiating AIFS and CW$_{\text{min}}$. However, note that differentiating TXOP may affect performance metrics not considered by our models, expecially delay jitter, since the traffic becomes more and more bursty.

6.8 Related work

Earlier work on the performance of IEEE 802.11 ad hoc networks considers a variety of scenarios, cf. [80] and the references therein. Also, [58] provides an extensive simulation study of the EDCA, investigating the impact of the various differentiation mechanisms on the performance for a single-hop network. However, these studies do not explicitly address the delays or throughputs in a *multihop* ad hoc network. The only paper we are aware of explicitly addressing an analytical evaluation of the multihop case, that is, the two-hop case we also address here, is [80]. In this

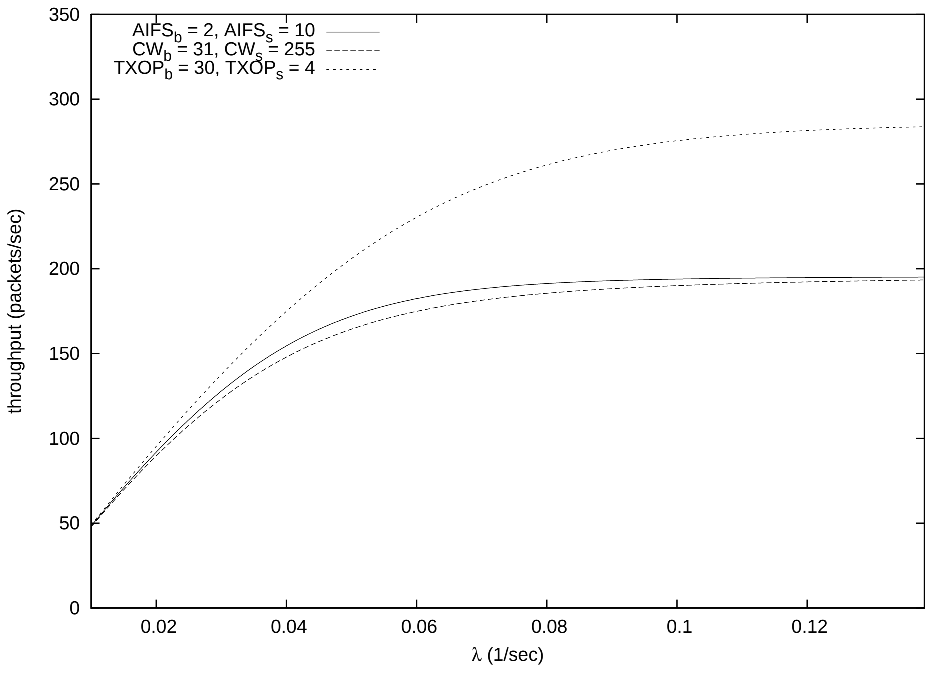


Figure 6.22: Maximum obtainable throughput per differentiation parameter as a function of λ

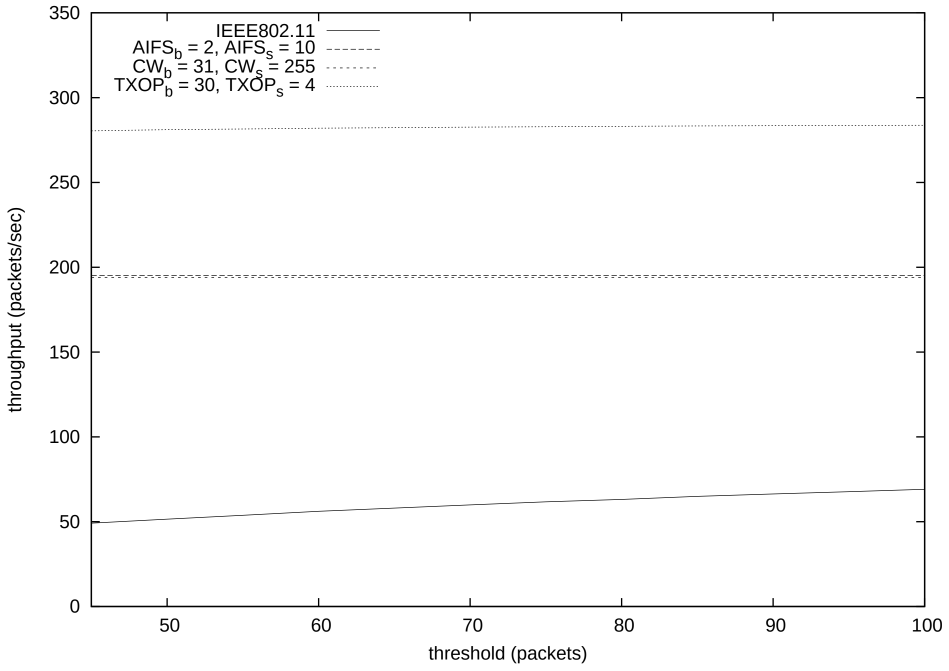


Figure 6.23: Maximum obtainable throughput per differentiation parameter for different bounds on the expected buffer occupancy

work the second hop has to forward the traffic of many sources (the first hops), thus forming a bottleneck, since all active stations have to share the transmission capacity. Explicit (closed-form) equations for the expected overall delay and the expected delay at the bottleneck, are obtained in [80] by translating the model at hand into a generalized processor sharing model, as studied by Cohen [24]. Although the analysis is approximate, good results are obtained, as confirmed by simulations. However, this evaluation approach is limited in that it only allows for an equal sharing of transmission capacity between all active stations (including the bottleneck). They do not address the differentiation parameters introduced in the protocol IEEE 802.11e.

6.9 Summary

We have presented a new model for analyzing the recently standardized quality-of-service enhancements of the IEEE 802.11e access mechanism in a two-hop ad hoc network. Our high-level model is flow-based, and uses results from packet-based models (such as those proposed by Bianchi and Engelstad et al. [14, 29]), and allows for the numerical evaluation of the buffer occupancy at the bottleneck node, the system throughput, as well as provides information on the mean number of active sources. The latter is important, as our model allows for a time-varying number of sources, as opposed to earlier models that only allow for a fixed number of sources. The model is very easy to use (and extensible) as the basic model structure remains the same for all enhancements; just two allocation functions (denoted as $S_b(\cdot)$ and $S_s(\cdot)$) need to be defined. An efficient numerical solution based on the underlying quasi-birth-death structure of the model is automatically provided.

We compare our results with extensive simulations (using Opnet) and show that our models provide very accurate results at almost negligible cost in comparison to the simulations. No other analytical models that allow for similar evaluations have been proposed so far.

Finally, our models show that all differentiation parameters can be used to allocate capacity in a better way between the bottleneck and the sources. However, the throughput of the bottleneck differs, depending on the differentiation method used. We have shown that the maximum throughput can be obtained with differentiating TXOP_b and TXOP_s. The resulting throughput is about 50% larger than when differentiating AIFS and CW_{min}.

Chapter 7

Conclusions

In this thesis, we have addressed stochastic model checking techniques for structured infinite Markov chains. Such Markov chains are very widely applicable, yet are structurally restricted enough to allow for efficient model checking. Our specific contributions to this research field are the following:

Model checking algorithms for two classes of infinite-state CTMCs

We presented model checking algorithms for the full range of CSL properties for the following two classes of infinite-state CTMCs:

- labeled quasi birth death processes and
- labeled Jackson queueing networks.

The model checking algorithms for the next operator and the until operator with its different intervals are based on the same approach for QBDs and JQNs. The only difference arises from the splitting of the infinite state space of QBDs into levels and the splitting of the infinite state space of JQNs into fronts. These model checking algorithms make extensive use of uniformization with representatives which is both computationally and memory efficient. Furthermore, we presented an optimal termination criterion for model checking the time bounded until operator.

Model checking the steady-state operator is done differently for QBDs and JQNs: on QBDs we use an iterative method based on the matrix geometric method and a termination criterion that is especially tailored for this algorithm. On JQNs the steady-state operator can be checked directly via a closed-form solution according to Jackson's theorem.

Except for the next operator in general and the steady-state operator on JQNs, the presented iterative algorithms compute approximations of the steady-state and the transient probabilities. Note that these algorithms do not stop in the very special case that the probability bound of the CSL formula is equal to one of the approximated probabilities. However, if the decision algorithm stops we can be sure that the validity of the CSL formula is stated correctly.

For both classes of infinite-state CTMCs we required independent atomic properties. Together with the splitting of the state-space, based on its special structure, this facilitated the use of representatives to cut the infinite state space to a finite representation. Note that the model checking algorithms presented in this thesis can be applied to every infinite state Markov chain for which such representatives can be found. We showed the feasibility of the presented model checking algorithms for QBDs and JQNs with a case study.

Extensions to QBDs and JQNs in the context of CSL model checking

We described several extensions of QBDs and JQNs for which model checking with the framework presented in this thesis is feasible as well as the extension toward CSRL model checking. However, for other extensions, e.g., tree-like QBDs and multi-class queueing networks, model checking with representatives is theoretically feasible, but not practically, as the underlying state space and the number of representatives becomes very large. To overcome this, further work will analyze the use of abstraction techniques.

IEEE 802.11e case study on bottlenecks in two-hop ad hoc networks

We presented a detailed case study on the analysis of bottlenecks in two-hop wireless ad hoc networks, validated by extensive simulation was provided.

The algorithms derived for model checking QBDs were applied to analyze bottlenecks in an IEEE 802.11e two-hop ad hoc network. The bottleneck was modeled as infinite stochastic Petri net into which the packet-level model by Engelstad et al. is incorporated. We provided a comprehensive comparison with detailed (packet-level) simulations using OPNET, showing very favorable results. Moreover, the analytical techniques are much faster than simulation, which allows us to compute the maximum obtainable throughput per parameter setting. Note that the two-hop scenario results in a QBD as there arises only one bottleneck. In a multi-hop network multiple bottlenecks may form and the underlying state space of such a network is not a QBD anymore. Further work will approach bottlenecks that occur in multi-hop networks, as well as multiple traffic classes.

Appendix A

Matrix-geometric method

The steady-state vector $\pi^{\mathcal{M}}$ is to be partitioned into sub vectors according to the different levels of QBD. The vector $b \in \mathbb{R}^{N_0}$ contains the steady-state probabilities of the boundary level and the vectors $v_i \in \mathbb{R}^N$ for $i = 0, 1, \ldots$ contain the steady-state probabilities of the i-th repeating level. Thus $\pi^{\mathcal{M}}$ can be rewritten as $\pi^{\mathcal{M}} = (b, v_0, v_1, \ldots)$.

Now, the equation for the steady-state probabilities, can be rewritten, exploiting the partitioning of $\pi^{\mathcal{M}}$ and the special structure of the generator matrix $\mathbf{Q}$. To avoid confusion the vector with all zeros is denoted $\overline{0}$.

$$\pi^{\mathcal{M}}\mathbf{Q} = \overline{0} \;\Leftrightarrow\; (b, v_0, v_1, v_2, \ldots) \cdot \begin{pmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} & & 0 & \\ \mathbf{B}_{1,0} & \mathbf{B}_{1,1} & \mathbf{A}_0 & & \\ & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \\ & & \mathbf{A}_2 & \mathbf{A}_1 & \ddots \\ 0 & & & \mathbf{A}_2 & \ddots \\ & & & & \ddots \end{pmatrix} = \overline{0} \tag{A.1}$$

$$\begin{aligned} & & b\mathbf{B}_{0,0} \quad + v_0\mathbf{B}_{1,0} &= 0 && \text{boundary level} && \text{(A.2)} \\ \Leftrightarrow & \wedge & b\mathbf{B}_{0,1} + v_0\mathbf{B}_{1,1} + v_1\mathbf{A}_2 &= 0 && \text{border level} && \text{(A.3)} \\ & \wedge & v_j\mathbf{A}_0 + v_{j+1}\mathbf{A}_1 + v_{j+2}\mathbf{A}_2 &= 0 && \text{repeating levels } j = 0, 1, \ldots && \text{(A.4)} \end{aligned}$$

under the normalization condition

$$b \cdot \overline{1} + \sum_{i=0}^{\infty} v_i \overline{1} = 1, \quad \text{with } \overline{1} = (1, 1, 1, \ldots). \tag{A.5}$$

With QBDs, the probability of residing in level i only depends on the probability of residence in level $i-1$ and level $i+1$ [38]. Since the transition rates between neighboring levels are constant, it is *assumed* that the steady-state vector follows the so called *matrix-geometric form*:

$$v_i = v_{i-1}\mathbf{R} = v_0\mathbf{R}^i, \qquad \text{with constant } \mathbf{R} \in \mathbb{R}^{N\times N}; \tag{A.6}$$

where for the time being the matrix **R** is unknown. With equation (A.6), (A.4) can be rewritten as follows:

$$\begin{aligned} & v_j\mathbf{A}_0 + v_{j+1}\mathbf{A}_1 + v_{j+2}\mathbf{A}_2 && = \overline{0} && \text{(A.7)}\\ \Leftrightarrow\quad & v_0\mathbf{R}^j\mathbf{A}_0 + v_0\mathbf{R}^{j+1}\mathbf{A}_1 + v_0\mathbf{R}^{j+2}\mathbf{A}_2 && = \overline{0} && \text{(A.8)}\\ \Leftrightarrow\quad & v_0\mathbf{R}^j(\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2) && = \overline{0} && \text{(A.9)} \end{aligned}$$

In order to fulfill equation (A.9) either $v_0\mathbf{R}^j$ or the term in parentheses needs to be zero. Note that $v_0\mathbf{R}^j = 0$ does not yield any useful solutions, because $v_0 = 0 \Rightarrow v_i = 0\ \forall i$ and $\mathbf{R} = 0 \Rightarrow v_i = 0\ \forall i$. It has been shown that **R** is the entry-wise smallest non-negative solution of:

$$\mathbf{R}^2\mathbf{A}_2 + \mathbf{R}^1\mathbf{A}_1 + \mathbf{R}^0\mathbf{A}_0 = \overline{0}. \tag{A.10}$$

Thus if we pick a matrix **R** such that (A.10) is fulfilled, assumption (A.4) is correct. The explicit computation of **R** from the matrix-quadratic equation (A.10) can be done with different well-known algorithms, for example:

- the successive substitution algorithm [61, 54],
- the logarithmic reduction [55].

Once the matrix **R** has been computed, the boundary probability vectors can be calculated using, for example, Gauss-elimination. Equations (A.2) and (A.3) can be written as matrix-vector products as follows:

$$(b, v_0)\begin{pmatrix}\mathbf{B}_{0,0}\\ \mathbf{B}_{1,0}\end{pmatrix} = \overline{0} \quad \text{and} \quad (b, v_0, v_1)\begin{pmatrix}\mathbf{B}_{0,1}\\ \mathbf{A}_1\\ \mathbf{A}_2\end{pmatrix} = \overline{0}. \tag{A.11}$$

Because of $v_1\mathbf{A}_2 = (v_0\mathbf{R})\mathbf{A}_2$, the second equation can be rewritten as

$$(b, v_0)\begin{pmatrix}\mathbf{B}_{0,1}\\ \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2\end{pmatrix} = \overline{0}, \tag{A.12}$$

which leads to the following equation

$$(b, v_0)\begin{pmatrix}\mathbf{B}_{0,0} & \mathbf{B}_{0,1}\\ \mathbf{B}_{1,0} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2\end{pmatrix} = (\overline{0}, \overline{0}), \tag{A.13}$$

that can be uniquely solved together with the normalization equation

$$b \cdot \overline{1} + (\sum_{j=0}^{\infty} \cdot v_j \overline{1}) = \overline{1}.$$

By using the geometric series the normalization equation can be rewritten under the condition that the largest eigenvalue of **R** is smaller than 1:

$$b \cdot \overline{1} + v_0 \cdot (\sum_{i=0}^{\infty} \mathbf{R}^i) \cdot \overline{1} = b \cdot \overline{1} + v_0 (\mathbf{I} - \mathbf{R})^{-1} \cdot \overline{1} = 1. \qquad \text{(A.14)}$$

As soon as the boundary vectors have been computed, the remaining steady-state vectors can be calculated by using recursion (A.6).

As stated above the geometric series can only be used, if the largest eigenvalue of **R** is smaller than 1. Otherwise the QBD is unstable and no steady-state probability can be derived. But there is still a more intuitive explanation for stability in QBDs that can be derived from true birth-death processes like $M|M|1$ queues. In $M|M|1$ queues, the steady-state probability only exists if the queue is stable, that is, if the drift to the next higher level (usually denoted λ) is smaller than the drift to the next smaller level (usually denoted μ), which leads to $\lambda < \mu$ and $\frac{\lambda}{\mu} = \rho < 1$. This result can be transfered to QBDs as follows.

A new matrix $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A}_2$ is interpreted as a generator matrix of a new CTMC. This CTMC resembles the QBD but transitions between levels are just led back to the same level. As **A** is finite, the steady-state probability ν of the new CTMC can be easily computed. Then the drift to the next higher level can be described as:

$$\sum_i v_i \sum_l \mathbf{A}_0(i,l) \quad = \quad v \cdot \mathbf{A}_0 \cdot \overline{1} \qquad \text{(A.15)}$$

and the drift to the next lower level as:

$$\sum_i v_i \sum_l \mathbf{A}_2(i,l) \quad = \quad v \cdot \mathbf{A}_2 \cdot \overline{1}. \qquad \text{(A.16)}$$

The requirement for stability and thus for the existence of the steady-state probability can be expressed as [38]: $\nu \mathbf{A}_0 \overline{1} < \nu \mathbf{A}_2 \overline{1}$.

Acknowledgments

Just like my choice to study computer science, my choice to delve into the area of performance evaluation has been rather random and mostly driven by my interest in mathematics. Back in 2001 at the RWTH when I started following the course Performance evaluation, lectured by Boudewijn Haverkort and tutored by Lucia Cloth, I had no idea that this very interesting and entertaining course would be the first step on my journey towards a PhD in the area of Model checking. I would not have been able to complete this journey without the aid and support of countless people.

First of all I would like to express my gratitude to my supervisor Boudewijn Haverkort, whose expertise, understanding and patience made this thesis possible. I am particularly grateful to Lucia Cloth for her motivation, encouragement and our fruitful technical discussions. Many thanks go to Geert Heijenk for his support and patience when fighting the network simulator. I thank all the reviewers for their comments on this thesis, special thanks go to Holger Hermanns and Markus Siegle for their very detailed comments.

Apart from my colleagues I would like to thank my parents, Gabriele and Reinhard Remke, who offered me unconditional love and support throughout my whole life, to them education has always been very important for my development. I must acknowledge Sido Harms for his support and patience during the last years and for his ability to deal with the stressed out version of myself.

Last but not least I would like to thank all the others who have made my stay here worthwhile.

Bibliography

[1] IEEE 802.11e/D13.0. Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) specifications: Medium Access Control Enhancements for Quality of Service (QoS). *IEEE 802.11 std.*, 2005.

[2] P.A. Abdulla, N.B. Henda, and R. Mayr. Verifying infinite Markov chains with a finite attractor or the global coarseness property. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 127–136. IEEE Press, 2005.

[3] P.A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A survey of regular model checking. In *Proc. 15th Int. Conference on Concurrency Theory (Concur'04)*, volume 3170 of *LNCS*, pages 35–48. Springer, 2004.

[4] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modeling with Generalised Stochastic Petri Nets.* John Wiley & Sons, 1995.

[5] A. Aziz, K. Sanwal, and R. Brayton. Model checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[6] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems: An algorithmic approach. In *Formal Methods for Real-Time and Probabilistic Systems (AMAST'99)*, volume 1601 of *LNCS*, pages 34–52. Springer, 1999.

[7] C. Baier, B.R. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *Proc. 27th Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 780–792. Springer, 2000.

[8] C. Baier, B.R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[9] C. Baier, B.R. Haverkort, H. Hermanns, and J.-P. Katoen. Comparative branching-time semantics for Markov chains. *Information and Computation*, 200(2):149–214, 2005.

[10] C. Baier, H. Hermanns, J.-P. Katoen, and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science*, 345:2–26, 2005.

[11] F. Baskett, K. Mani Chandy, R.R. Muntz, and R.G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.

[12] A. Bell. *Distributed evaluation of stochastic Petri nets.* PhD thesis, RWTH Aachen, 2004.

[13] N. Bertrand and Ph. Schnoebelen. Model checking lossy channel systems is probably decidable. In *Proc. 6th Foundations of Software Science and Computation Structures (FOSSACS'03)*, volume 2060 of *LNCS*, pages 120–135. Springer, 2003.

[14] G. Bianchi. Performance analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18:535–547, 2000.

[15] D.D. Bini, G. Latouche, and B. Meini. Solving nonlinear matrix equations arising in Tree-Like stochastic processes. *Linear Algebra and its Applications*, 366:39–64, 2003.

[16] T. Brázdil, A. Kucera, and O. Strazovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proc. 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, volume 3404 of *LNCS*, pages 145–157. Springer, 2005.

[17] P. Buchholz. A class of hierarchical queueing systems and their analysis. *Queueing Systems*, 15:59–80, 1994.

[18] P. Buchholz, J.-P. Katoen, P. Kemper, and C. Tepper. Model checking large structured Markov chains. *Journal of Logic and Algebraic Programming*, 56(1-2):69–97, 2003.

[19] G. Ciardo. Discrete-time Markovian stochastic Petri nets. In *Computations with Markov Chains*, pages 339–358. Raleigh, 1995.

[20] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[21] L. Cloth. *Model checking algorithms for Markov Reward Models.* PhD thesis, University of Twente, 2006.

[22] L. Cloth, J.-P. Katoen, M. Khattri, and R. Pulungan. Model checking Markov reward models with impulse rewards. In *Proc. Int. Conference on Dependable Systems and Networks (DSN'05)*, pages 722–731. IEEE Press, 2005.

[23] J.W. Cohen. *The single server queue.* North-Holland, 1969.

[24] J.W. Cohen. The multiple phase service network with Generalized Processor Sharing. *Acta informatica*, 12:245–284, 1979.

[25] A.E. Conway and N.D. Georganas. *Queueing Networks: Exact Computational Analysis.* MIT Press, 1989.

[26] D.R. Cox. A use of complex probabilities in the theory of stochastic processes. *Proc. Cambridge Phil. Soc.*, 51:313–319, 1955.

[27] D. D'Aprile, S. Donatelli, and J. Sproston. CSL Model Checking for the Great-SPN Tool. In *Proc. 19th Int. Symposium on Computer and Information Sciences (ISCIS'04)*, volume 3280 of *LNCS*, pages 543–552. Springer, 2004.

[28] M.B. Dwyer, G.S. Avrunin, and J.C. Corbett. Patterns in property specification for finite-state verification. In *Proc. 21st Int. Conference on Software Engineering (ICSE'99)*, IEEE Press, pages 411–420, 1999.

[29] P.E. Engelstad and O.N. Osterbo. Non-saturation and saturation analysis of IEEE 802.11e EDCA with starvation prediction. In *Proc. 8th ACM Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '05)*, pages 224–233. ACM Press, 2005.

[30] J. Esparza, A. Kucera, and R. Mayr. Model checking probabilistic pushdown automata. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 12–21. IEEE Press, 2004.

[31] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *Proc. 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, volume 3404 of *LNCS*, pages 340–352. Springer, 2005.

[32] H. Fecher, M. Leuker, and V. Wolf. Don't Know in Probabilistic Systems. In *Model Checking Software*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.

[33] W.K. Grassmann. Transient solution in Markovian queueing systems. *Computations in Operations Research*, 4:47–56, 1977.

[34] W.K. Grassmann. Finding transient solutions in Markovian event systems through randomization. In *Numerical solution of Markov chains*, pages 411–420. Dekker, 1991.

[35] W.P. Groenendijk. *Conservation Laws in Polling Systems.* PhD thesis, University of Utrecht, 1990.

[36] D. Gross and D.R. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):343–361, 1984.

[37] P.G. Harrison. Transient behaviour of queueing networks. *Journal of Applied Probability*, 18(2):482–490, 1981.

[38] B.R. Haverkort. *Performance of Computer Communication Systems.* John Wiley and Sons, 1999.

[39] B.R. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model checking performability properties. In *Proc. Int. Conference on Dependable Systems and Networks (DSN'02)*, pages 102–112. IEEE Press, 2002.

[40] B.R. Haverkort and A. Ost. Steady-State Analysis of Infinite Stochastic Petri Nets: Compering Between the Spectral Expansion and the Matrix-Geometric Method. In *Proc. 7th Int. Workshop on Petri Nets and Performance Models (PNPM'97)*. IEEE Press, 1997.

[41] G.J. Heijenk and B.R. Haverkort. Design and evaluation of a connection management mechanism for an ATM-based connectionless service. *Distributed System Engineering Journal*, 3(1):53–67, 1996.

[42] H. Hermanns, U. Herzog, and J.P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1-2):43–87, 2002.

[43] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *International Journal on Software Tools for Technology Transfer*, 4(2):153–172, 2003.

[44] J. Hillston. *A Compositional Approach to Performance Modeling.* Camebridge University Press, 1996.

[45] S.P. Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proc. Theory and Practice of Software Development (TAPSOFT'97)*, volume 1214 of *LNCS*, pages 667–681, 1997.

[46] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5(4):518–521, 1957.

[47] J.R. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.

[48] J.-P. Katoen, M. Khattri, and I.S. Zapreev. A Markov Reward Model Checker. In *Proc. 3rd Int. Conference on the Quantitative Evaluation of Systems (QEST'06)*, pages 243–244. IEEE press, 2005.

[49] J.-P. Katoen, D. Klink, M. Leuker, and V. Wolf. Three-Valued Abstraction for Continuous-Time Markov Chains. In *Proc. 19th Int. Conference on Computer Aided Verification (CAV'07)*, volume 4590 of *LNCS*, pages 311–324. Springer, 2007.

[50] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *Proc. 9th Int. Conference on Computer Aided Verification (CAV'97)*, volume 1254 of *LNCS*, pages 424–435. Springer, 1997.

[51] L. Kleinrock. *Queueing Systems; Volume 1: Theory.* John Wiley & Sons, 1975.

[52] J.F. Kurose and K.W. Ross. *Computer Networking.* Addison-Wesley, 2005.

[53] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[54] G. Latouche. Algorithms for infinite Markov chains with repeating columns. *Linear algebra, Markov chains, and queueing models*, 48:231–265, 1993.

[55] G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi birth and death processes. *Journal of Applied Probability*, 30:650–674, 1993.

[56] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* ASA-SIAM, 1999.

[57] L.M. Le Ny and B. Sericola. Transient analysis of the BMAP/PH/1 queue. *International Journal of Simulation*, 3(3-4):4–15, 2003.

[58] R. Litjens, R. Roijers, J.L. van den Berg, R.J. Boucherie, and M.J. Fleuren. Analysis of flow transfer times in IEEE 802.11 wireless lans. *Annals of Telecommunications*, 59:1407–1432, 2004.

[59] D.A. Menasce, V.A.F. Almeida, and L.W. Dowdy. *Performance by Design.* Prentice Hall, 2004.

[60] I. Mitrani and R. Chakka. Spectral expansion solution for a class of Markov models: Application and comparison with the matrix-geometric method. *Performance Evaluation*, 23(3):241–260, 1995.

[61] M.F. Neuts. *Matix Geometric Solutions in Stochastic Models: An Algorithmic Approach.* Johns Hopkins University Press, 1981.

[62] Opnet modeler software. available: http://www.opnet.com/products/modeler.

[63] A. Ost. *Performance of Communication Systems. A Model-Based Approach with Matrix-Geometric Methods*. PhD thesis, RWTH Aachen, 2001.

[64] A.M. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. In *Proc. 5th Int. Conference on Formal Modelling and Analysis of Timed Systems Automata, Languages and Programming, (ICALP'03)*, volume 2719 of *LNCS*, pages 1008–1021. Springer, 2003.

[65] A. Remke. Model Checking Quasi Birth Death Processes. Master's thesis, RWTH Aachen, 2004.

[66] A. Remke and B.R. Haverkort. Beyond Model-Checking CSL for QBDs: Resets, Batches and Rewards. In *Proc. 7th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS'05)*, pages 23–27, 2006.

[67] A. Remke and B.R. Haverkort. CSL model checking algorithms for infinite-state structured Markov chains. In *Proc. 5th Int. Conference on Formal Modelling and Analysis of Timed Systems (Formats'07)*, volume 4763 of *LNCS*, pages 336–351. Springer, 2007.

[68] A. Remke, B.R. Haverkort, and L. Cloth. Model checking infinite-state Markov chains. In *Proc. 11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, volume 3440 of *LNCS*, pages 237–252. Springer, 2005.

[69] A. Remke, B.R. Haverkort, and L. Cloth. Bottlenecks in Two-Hop Ad Hoc Networks: Dividing Radio Capacity in a Smart Way. In *Workshop on Stochastic Performance Models for Resource Allocation in Communication Systems (STOPERA'06)*, pages 23–26, 2006.

[70] A. Remke, B.R. Haverkort, and L. Cloth. Uniformization with Representatives. In *Proc. Workshop on Tools for solving structured Markov chains*. ACM press (CD only), 2006.

[71] A. Remke, B.R. Haverkort, and L. Cloth. A versatile infinite-state Markov reward model to study bottlenecks in 2-hop ad hoc networks. In *Proc. 3rd Int. Conference on the Quantitative Evaluation of Systems (QEST'06)*, pages 63–72. IEEE Press, 2006.

[72] A. Remke, B.R. Haverkort, and L. Cloth. CSL model checking algorithms for QBDs. *Theoretical Computer Science*, 2007.

[73] A. Remke, B.R. Haverkort, G. Heijenk, and L. Cloth. Bottleneck Analysis for Two-Hop IEEE 802.11e ad hoc Networks. In *Proc. 15th Int. Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'08)*, volume 5055 of *LNCS*, pages 279–294. Springer, 2008.

[74] A. Riska. *Aggregate matrix-analytic techniques and their applications.* PhD thesis, The College of William and Mary in Virginia, 2002.

[75] F. Roijers, J.L. van den Berg, X. Fan, and M. Fleuren. A performance study on service integration in IEEE 802.11e wireless LANs. *Computer Communications*, 29:2621–2633, 2006.

[76] J. Schiller. *Mobile Communications.* Addison-Wesley, 2003.

[77] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[78] T. Takine, B. Sengupta, and R. W. Yeung. A Generalization of the Matrix M/G/1 Paradigm for Markov Chains with a Tree Structure. *Communications in Statistics - Stochastic Models*, 11(3):411–421, 1995.

[79] K.S. Trivedi. *Probability and Statistics with Reliability, Qeueing and Computer Science Applications.* John Wiley & Sons, 2002.

[80] H. van den Berg, M. Mandjes, and F. Roijers. Performance modeling of a bottleneck node in an IEEE 802.11 ad-hoc network. In *Proc. 5th Int. Conference on AD-HOC Networks and Wireless (AdHoc-NOW'06)*, volume 4104 of *LNCS*, pages 321–336. Springer, 2006.

[81] B. van Houdt and C. Blondia. Tree Structured QBD Markov Chains and Tree-Like QBD Processes. *Stochastic Models*, 19(4):467– 482, 2003.

[82] B. van Houdt and C. Blondia. Approximated transient queue length and waiting time distributions via steady-state analysis. *Stochastic Models*, 21:725–744, 2005.

[83] B. van Houdt and C. Blondia. QBDs with marked time epochs: a framework for transient performance measures. In *Proc. 2nd Int. Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 210–219. IEEE press, 2005.

[84] A.P.A. van Moorsel. *Performability Evaluation Concepts and Techniques.* PhD thesis, University Twente, 1993.

[85] J. van Velthoven, B. van Houdt, and C. Blondia. Transient analysis of tree-like processes and its application to random access systems. *ACM SIGMETRICS Performance Evaluation Review*, 34(1):181–190, 2006.

[86] J. van Velthoven, B. van Houdt, and C. Blondia. Simultaneous transient analysis of QBD Markov chains for all initial configurations using a level-based recursion. In *Proc. 4th Int. Conference on the Quantitative Evaluation of Systems (QEST'07)*, pages 79–88. IEEE press, 2007.

[87] R. W. Yeung and B. Sengupta. Matrix Product-Form Solutions for Markov Chains with a Tree Structure. *Advances in Applied Probability*, 26(4):965–987, 1994.

[88] R.W. Yeung and A.S. Alfa. The Quasi-Birth-Death Type Markov Chain with A Tree Structure. *Communications in Statistics-Stochastic Models*, 15(4):639–659, 1999.

[89] J. Zhang and E.J. Coyle. Transient analysis of quasi-birth-death processes. *Stochastic Models*, 5(3):459–496, 1989.

[90] L. Zhang, H. Hermanns, E. M. Hahn, and B. Wachter. Time-bounded model checking of infinite-state continuous-time Markov chains. In *Proc. 8th Int. Conference on Application of Concurrency to System Design (ACSD'08)*, to be published, 2008.