Solving Infinite Stochastic Process Algebra Models Through Matrix-Geometric Methods

Amani El-Rayes, Marta Kwiatkowska and Gethin Norman

University of Birmingham, Birmingham B15 2TT, UK
{ahe,mzk,gxn}@cs.bham.ac.uk

Abstract. We introduce a Stochastic Process Algebra called $\text{PEPA}^{\infty}_{\text{ph}}$, based on Hillston's PEPA. $\text{PEPA}^{\infty}_{\text{ph}}$ is suitable for describing and analysing the performance of certain kinds of queues, such as $Ph/Ph/c$ and $M/Ph/1$. The activities of $\text{PEPA}^{\infty}_{\text{ph}}$ components have durations given by *phase-type distributions*. To overcome the state space explosion that arises when solving the models through the underlying Markov process we instead use the *Matrix-Geometric Method*. Though the method proposed here can only be applied to a fragment of $\text{PEPA}^{\infty}_{\text{ph}}$ because of its dependence on the structure of the system, we can solve models with potentially infinitely many customers queued (an unbounded buffer), in contrast to the approaches used in SPAs such as PEPA, TIPP and EMPA.

1 Introduction

Conventional Stochastic Process Algebras, to mention PEPA [1], only allow durations of activities to be exponentially distributed. Though this has several advantages, such as a straightforward representation for distributions (the duration of an activity can be simply represented by a single real number denoting the rate of the activity) and efficiency of calculation (performed through the single-parameter functional form without the need for tabulation), the restriction on the duration to be exponentially distributed is often unrealistic. Many systems, e.g. $M/E_n/1$ queues, require distributions other than exponential. Another difficulty is the rapid increase in the size of the state space of the underlying Markov process (known as the state space explosion problem). We attempt to overcome these problems by formulating a Stochastic Process Algebra based on PEPA [1], called $\text{PEPA}^{\infty}_{\text{ph}}$, which allows us to specify systems with potentially *infinitely many customers* in an infinite queueing system with durations given as *phase-type* distributions, and calculating the steady-state probability of the models that arise from a fragment of $\text{PEPA}^{\infty}_{\text{ph}}$ through the Matrix-Geometric Method (MGM). Thus, the MGM enables us to avoid the explicit construction of the generator matrix of the underlying Markov chain (hence tackling the state space explosion problem), while at the same time allowing us to consider systems with potentially infinitely many customers (hence also infinitely many states if fully unfolded), combined with the generality that phase-type distributions may offer.

J. Hillston and M. Silva (Eds.), 7th Process Algebras and Performance Modelling Workshop (PAPM'99), pages 41–62, University of Zaragoza, 1999.

The queueing systems we can analyze in $\text{PEPA}^{\infty}_{\text{ph}}$ are scheduled according to *first in-first out* and have structure including the following classes:

1. queueing models with a *Quasi-Birth-Death* structure: in these queues the transitions can only happen between the adjacent levels/groups of states in the model,
2. queueing models *with failure*, subject to the restriction that if a processor fails, all the customers in the system are assumed to be lost and all other transitions only happen between the adjacent levels/groups of states,
3. a *bulk service* queue, in which the server can serve up to k customers at a time and all other transitions only happen between the adjacent levels/groups of states.

We allow durations of activities to be given in terms of phase-type distributions as defined by Neuts [2], which generalises the conventional approach employing only exponentials. In contrast to exponential distributions (which are closed under minimum only), the class of phase-type distributions has very strong closure properties: they are closed under maximum, minimum and convolution. This allows us to define the *synchronisation*, *choice* and *prefixing* of two $\text{PEPA}^{\infty}_{\text{ph}}$ components respectively as the *maximum*, *minimum* and *convolution* of the phase-type distributions of their activities. The activities in the composed system thus have phase-type distributed durations. Another advantage of phase-type distributions is that any other distribution defined on the interval $[0, \infty)$ can be approximated by arbitrarily accurate phase-type distributions [3].

The Matrix-Geometric Method (see e.g. [3,4,2]) relies on identifying two portions (parts) within the structure of the underlying Continuous Time Markov Chain (state transition system): the initial portion and the repetitive portion. The *initial* portion (which must be finite) has a non-regular structure and each component in it must be represented in detail. The repetitive portion has a regular structure and can be represented in SPAs as a composition of several components. In MGM the generator matrix is decomposed into submatrices, with each one of them representing the transition rates in a particular area within a given portion, or between them. The size of the state space in MGM, even if the system is infinite, is reasonably small compared with the size of the generator matrix of the Markov process. Through solving the system of submatrices by using MGM we obtain the steady-state probability, which we can use to further compute the performance measures of the model.

The mainstream SPAs (PEPA [1], TIPP [5] and EMPA [6]) are extensions of CCS that allow only exponentially distributed durations and differ in the approach to synchronisation. There has been recent work extending SPAs to allow general distributions including [7,8,9] and [10]. Furthermore, in [11] an approach for analysing the performance of a plain-old telephone system is introduced by defining a stochastic process algebra that separates the advance of time and action occurrences, where time can be described by a continuous phase-type distribution. The induced state space is very large, and semantic equivalence checking is introduced to minimise the size of the generator matrix. The Matrix-Geometric Method is used in [12] to solve infinite stochastic Petri Nets which

have an underlying Markov chain of the *Quasi-Birth-Death* type. In [13] a multi-server queue with non-preemptive heterogeneous priority structures is analyzed and the MGM is used to derive the stationary distribution of the queue and waiting times.

The work of Mitrani, Ost and Rettelbach [14] has similarities with the approach presented here: both modify SPAs to model queueing systems with potentially infinitely many customers queued. Our approach differs in that we extend PEPA as opposed to TIPP; the two differ as far synchronization is concerned. In addition, we allow phase-type distributions and solve the model through MGM, whereas in [14] only exponential distributions are used in conjunction with the spectral expansion method of [15].

The organisation of the paper is as follows. The next section gives a brief overview of phase-type distributions and their properties. In Section 3 the Matrix-Geometric Method is presented, including the calculation of steady-state probabilities and some performance measures. In Section 4 we introduce the process language $\text{PEPA}^{\infty}_{\text{ph}}$, its syntax and operational semantics. Section 5 describes the fragment of $\text{PEPA}^{\infty}_{\text{ph}}$ for which the steady-state probabilities can be calculated via MGM and the algorithm for deriving the submatrices for $\text{PEPA}^{\infty}_{\text{ph}}$ components in this class. The last section includes the conclusion and further work.

2 Phase-type distributions

We assume familiarity with Markov processes and probability theory, see e.g. [3].

The family of phase-type distributions is widely used in algorithmic probability. The advantage of phase-type distributions is their generality and versatility, which permits the calculation of the performance measures of stochastic models with a high degree of accuracy. Combined with the use of the Matrix Geometric Method [2], the exact and detailed characteristics of most of these models can be obtained relatively easily.

We now overview phase-type distributions following [2,3]. Consider an $(m+1)$-state Markov process. Assume that states $1, 2, \ldots, m$ are transient and that state $m+1$ is an absorbing state. Define $\alpha_i, i = 1, 2, \ldots, m+1$, to be the probability that the process is started in state i and define $\underline{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_m]$ (a row vector).

Define $\mathbf{T}_{ij} \geq 0$, for $i \neq j$ and $i \geq 1$, $j \leq m$, to be the rate of transition from state i to state j. T^0_i denotes the rate of transition from state i to the absorbing state $m+1$, $\underline{T}^0$ the column vector of these rates and $\mathbf{T}_{ii}$ is defined by $\mathbf{T}_{ii} = -T^0_i - \sum_{i \neq j} \mathbf{T}_{ij}$, for $1 \leq i \leq m$.

If $F(x)$ is the probability distribution of the time to absorption in the $(m+1)$-state with the infinitesimal generator

$$\mathbf{Q} = \begin{pmatrix} \mathbf{T} & \underline{T}^0 \\ \mathbf{0} & \underline{0} \end{pmatrix} \tag{1}$$

and initial probability vector $[\underline{\alpha}\ \alpha_{m+1}]$, then the pair $(\underline{\alpha}, \mathbf{T})$ is a representation of $F(x)$. Furthermore, the probability distribution $F(x)$ and the associated density function $f(x)$ are given by:

$$F(x) = 1 - \underline{\alpha}\ \exp(\mathbf{T}x)\ \underline{e} \quad \text{and} \quad f(x) = \underline{\alpha}\ \exp(\mathbf{T}x)\ \underline{T}^0 \quad \text{for } x \geq 0$$

respectively, where $\underline{e}$ is a column vector of 1's.

We consider the matrix $\mathbf{T}+\underline{T}^0\underline{\alpha}$ to be irreducible; then $(\underline{\alpha}, \mathbf{T})$ is an irreducible representation.

Definition 1. *A* continuous time phase-type distribution *(Ph-distribution) is the distribution of the time until absorption in an absorbing Markov process.*

Erlang (i.e. convolutions of identical exponential distributions), hyper-exponential, hypo-exponential and Coxian distributions are examples of phase-type distributions [2].

Phase-type distributions have many important properties [16] that make them useful in the evaluation of performance models: they are closed under *convolution*, *maximum* and *minimum*.

We shall require the following definition of a Ph-distribution resulting from a Ph-distribution being selected in a state with probability $r \in (0, 1)$.

Definition 2. *For any phase-type distribution $F(\bullet)$ with representation $(\underline{\alpha}, \mathbf{T})$ and $r \in (0, 1)$, let $r \cdot F(\bullet)$ denote the phase-type distribution with representation $(\underline{\alpha}, r \cdot \mathbf{T})$.*

3 Matrix-Geometric Methods

3.1 Introduction

The Matrix-Geometric Method [16,3,4] allows us to deal with the models whose activities are not necessarily exponentially distributed, while at the same time overcoming the problem of the rapid growth of the state space introduced by the need to explicitly construct the generator matrix of the underlying Markov process.

The MGM can only be applied if the system can be decomposed into two parts: the initial portion and the repetitive portion. For example, queueing systems with possibly an *unbounded number of customers queued* have such a structure: typically there exists an integer $i^\star$ such that from that $i^\star$ onwards the behaviour of the system for all $i \geq i^\star$ (where i is the number of queued customers) is the same as the behaviour of the system for $i^\star$. Such similarity need not hold for $0, 1, \ldots, i^\star - 1$. Thus we can represent the system by storing the information for the initial (also called *boundary*) portion $0, 1, \ldots, i^\star - 1$ and the repeating, or repetitive, portion $(i^\star, i^\star + 1 \ldots)$.

More precisely, consider a M/Ph/1 queueing model, where the *arrival rate is exponentially distributed* and the *service rate belongs to phase-type distributions*. We construct a Markov process (called a *vector-state process*) to solve this model

as follows. The (infinitely many) states are represented as pairs (i, j) where $i \geq 0$ is an unbounded integer representing the *number of queued customers*, and $j, 1 \leq j \leq m$ is the *phase* of the service. The set of states $\{(i,1),(i,2),\ldots,(i,m)\}$, for $i \geq 0$ is called *the level* i of the system. Then, there exists an integer $i^\star$ such that the levels 0 up to $i^\star - 1$ form the boundary, and those for $i \geq i^\star$ are repeating. Transitions between the repeating states have the property that the rates from state (i, j) to state $(i + k, j')$ for $0 \leq k < \infty$ and $0 \leq j, j' \leq m$, are independent of the value of i for $i \geq i^\star$. For simplicity, we let the initial portion (levels $0, \ldots, i^\star - 1$ as defined above) to be represented by level zero, and for the repetitive portion (levels $i^\star, i^\star + 1, \ldots$) by levels $1, 2, \ldots$ The repetition of the state transitions for vector processes implies a geometric form where scalars are replaced by matrices. Such Markov Processes are called Matrix-Geometric Processes.

3.2 Matrix-Geometric Solution

The theory of Matrix-Geometric Solutions was developed by Neuts [2] to solve the stationary state probabilities for the vector state Markov processes with repetitive structure.

Consider the generator matrix $\mathbf{Q}$ of a continuous time Markov process with the structure shown below:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B_{0,0}} & \mathbf{B_{0,1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{B_{1,0}} & \mathbf{B_{1,1}} & \mathbf{A_0} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{B_{2,0}} & \mathbf{B_{2,1}} & \mathbf{A_1} & \mathbf{A_0} & \mathbf{0} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

The matrix $\mathbf{Q}$ is composed of submatrices: $\mathbf{B_{0,0}}$, a square matrix of dimension $(m_1 - m)$; $\mathbf{B_{0,1}}$, a matrix of dimension $(m_1 - m) \times m$; $\mathbf{B_{k,0}}$, where $k \geq 1$, matrices of dimension $m \times (m_1 - m)$; with all the remaining submatrices square with dimension m, where $m_1 \geq m$.

The off-diagonal elements of $\mathbf{Q}$ are non-negative. The diagonal elements are all strictly negative and the row sums of $\mathbf{Q}$ equal zero.

The 0^{th} level of the process associated with $\mathbf{Q}$ are the $m_1 - m$ boundary states. The first level are the next m states, and likewise for the second level, and so on.

For simplicity, we will only consider the generator matrix $\mathbf{Q}$ with the following structure, where $\mathbf{B_{k,1}} = \mathbf{A_k}$ for all $k \geq 1$:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B_{0,0}} & \mathbf{B_{0,1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{B_{1,0}} & \mathbf{A_1} & \mathbf{A_0} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{B_{2,0}} & \mathbf{A_2} & \mathbf{A_1} & \mathbf{A_0} & \mathbf{0} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{B_{k,0}} & \mathbf{A_k} & \mathbf{A_{k-1}} & \mathbf{A_{k-2}} & \mathbf{A_{k-3}} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots \end{pmatrix}$$

Each submatrix contains the transition rates for a particular area. In the initial portion:

- $\mathbf{B_{0,0}}$ contains the transition rates within level 0
- $\mathbf{B_{0,1}}$ contains the transition rates from level 0 to level 1
- $\mathbf{B_{k,0}}$ contains the transition rates from level k in the repetitive portion to level 0.

In the repetitive portion:

- $\mathbf{A_0}$ contains the transition rates from level i to level $i+1$, for $i \geq 1$
- $\mathbf{A_1}$ contains the transition rates within level i, for $i \geq 1$
- $\mathbf{A_k}$ contains the transition rates from level i to level $i-k$, for $i > k \geq 1$.

Let $\underline{\pi}_i = [\pi_{i,1}, \pi_{i,2}, \ldots, \pi_{i,m}]$ for $i \geq 1$, $\underline{\pi}_0 = [\pi_{0,0}, \pi_{0,1}, \ldots, \pi_{0,m_1-m-1}]$, and $\underline{\pi} = [\underline{\pi}_0, \underline{\pi}_1, \underline{\pi}_2, \ldots]$.

The equation for the repeating states of the process is given (in block matrix form) by:

$$\sum_{k=0}^{\infty} \underline{\pi}_{j-1+k} \cdot \mathbf{A_k} = \underline{0} \text{ for } j = 1, 2, \ldots$$

The value of $\underline{\pi}_j$ (vector of stationary probabilities) is a function only of the transition rates between states with $j-1$ queued customers and states with j queued customers. Since these transition rates do not depend on j, then there is a constant matrix $\mathbf{R}$ such that:

$$\underline{\pi}_j = \underline{\pi}_{j-1} \cdot \mathbf{R} \text{ for } j = 1, 2, \ldots$$

and the values of $\underline{\pi}_j$ for $j = 1, 2, \ldots$ have the matrix-geometric form:

$$\underline{\pi}_j = \underline{\pi}_1 \cdot \mathbf{R}^{j-1}. \tag{2}$$

Simplification yields:

$$\sum_{k=0}^{\infty} \mathbf{R}^k \cdot \mathbf{A_k} = \mathbf{0}.$$

Solving this matrix polynomial gives the solution matrix $\mathbf{R}$ and then we can calculate the stationary probabilities for the repeating states. The matrix $\mathbf{R}$ is called *the rate matrix* and it is the *minimal* solution of the matrix polynomial.

For the initial portion we have:

$$\underline{\pi}_0 \cdot \mathbf{B_{0,0}} + \underline{\pi}_1 \cdot \mathbf{B_{0,1}} = \underline{0}$$

and

$$\sum_{k=1}^{\infty} \underline{\pi}_{k-1} \cdot \mathbf{B_{k,0}} + \sum_{k=1}^{\infty} \underline{\pi}_k \cdot \mathbf{A_k} = \underline{0}$$

Now using (2) we can rewrite the above in matrix form as:

$$(\underline{\pi}_0, \underline{\pi}_1) \begin{pmatrix} \mathbf{B_{0,0}} & \mathbf{B_{0,1}} \\ \sum_{k=1}^{\infty} \mathbf{R}^{k-1} \cdot \mathbf{B_{k,0}} & \sum_{k=1}^{\infty} \mathbf{R}^{k-1} \cdot \mathbf{A_k} \end{pmatrix} = \mathbf{0}. \tag{3}$$

This equation is not sufficient to determine the probabilities $\underline{\pi}_0$ and $\underline{\pi}_1$, so using the normalisation condition we obtain:

$$\underline{\pi}_0 \cdot \underline{e} + \left(\underline{\pi}_1 \cdot \sum_{k=1}^{\infty} \mathbf{R}^{k-1} \right) \cdot \underline{e} = \underline{\pi}_0 \cdot \underline{e} + \left(\underline{\pi}_1 \cdot (\mathbf{I} - \mathbf{R})^{-1} \right) \cdot \underline{e} = 1,$$

where $\underline{e}$ is a column vector of $1's$. The above equation together with equation (3) yields a unique solution. For any matrix $\mathbf{B}$ define the matrix $\mathbf{B}^*$ to be the matrix $\mathbf{B}$ with its first column eliminated and let the row vector $[1, \underline{0}]$ consist of a 1 followed by a suitable number of zeros, then the solution for the boundary states can be given by solving:

$$(\underline{\pi}_0, \underline{\pi}_1) \begin{pmatrix} \underline{e} & \mathbf{B_{0,0}}^* & \mathbf{B_{0,1}} \\ (\mathbf{I} - \mathbf{R})^{-1} \cdot \underline{e} & \left[\sum_{k=1}^{\infty} \mathbf{R}^{k-1} \cdot \mathbf{B_{k,0}} \right]^* & \sum_{k=1}^{\infty} \mathbf{R}^{k-1} \cdot \mathbf{A_k} \end{pmatrix} = [1, \underline{0}].$$

The above method can also be applied (approximately) to the case of finite models, as outlined in [12]. Under many circumstances, this method provides a good approximation for the steady-state probabilities of a finite model. When certain quasi-reversibility properties hold, the solution is even exact.

3.3 Computation of R

Assume that the generator matrix is irreducible. The necessary condition for this is that the matrices $\mathbf{B_{0,0}}$ and $\mathbf{A_1}$ are nonsingular, which means that we can calculate the inverse of the matrices.

The computation of the matrix $\mathbf{R}$ is by means of the iterative procedure. Start with the initial iteration $\mathbf{R}(0) = \mathbf{0}$, and calculate successive approximations as:

$$\mathbf{R}(n+1) = -\sum_{j=0, j\neq 1}^{\infty} \mathbf{R}^j(n) \cdot \mathbf{A_j} \cdot {\mathbf{A_1}}^{-1}, \quad n = 0, 1, \ldots$$

The sequence $\{\mathbf{R}(n)\}_n$ is entry-wise nondecreasing and converges monotonically to a nonnegative matrix $\mathbf{R}$. This follows from the fact that $-\mathbf{A}_1^{-1}$ is a nonnegative matrix. The number of iterations needed for convergence increases as the spectral radius of $\mathbf{R}$ increases.

We terminate the iteration and return with the solution of $\mathbf{R}$ when

$$\| \mathbf{R}(n+1) - \mathbf{R}(n) \|_{\infty} \leq \varepsilon,$$

where ε is a given small constants.

In many cases the matrices $\mathbf{A}_k$ are all zero for some k satisfying $k \geq 3$.

We note that the above algorithm for computing $\mathbf{R}$ can often converge very slowly, and hence a large number of iterative steps may be required. However, in the restricted case of Quasi-Birth-Death processes, we can greatly reduce the number of iterations through using the algorithm of Latouche and Ramaswami [17] for calculating $\mathbf{R}$ (see for example [4,18]).

3.4 Calculating performance measures

Having calculated the matrix $\mathbf{R}$, we can compute a range of performance measures for the model directly from $\mathbf{R}$. For example:

- the expected number N of queued customers:

$$N = \underline{\pi}_1 \cdot (\mathbf{I} - \mathbf{R})^{-2} \cdot \underline{e}$$

- the conditional probabilities $\{q_k(j)\}$ of being in phase j given the system is at level k, for $k \geq 0$ and $1 \leq j \leq m$:

$$q_k(j) = (\underline{\pi}_j^*)^{-1} \cdot \pi_{kj}$$

- the probability J_k of having k jobs in the queue, for $k \geq 1$:

$$J_k = \underline{\pi}_1 \cdot \mathbf{R}^{k-1} \cdot (\mathbf{I} - \mathbf{R})^{-1} \cdot \underline{e}.$$

4 The Process Language

We now introduce our process language based on PEPA [1]. We call this language $\text{PEPA}^{\infty}_{\text{ph}}$, where the superscript ∞ denotes that we can model queues with an unbounded number of customers and *ph* stands for phase-type distributions.

4.1 Syntax and Operational Semantics

In $\text{PEPA}^{\infty}_{\text{ph}}$ an activity is represented by a pair (a, ph), where $a \in \mathcal{A}ct$, $\mathcal{A}ct$ is a countable set of all possible action types, and ph is a random variable representing the duration of a. We denote by τ a distinguished symbol for internal activity not included in $\mathcal{A}ct$. We assume that ph belongs to the class of *phase-type distributions* and is represented by $(\underline{\alpha}, \mathbf{T})$ as described in Section 2, or $ph = \top$ denoting a *passive* activity of [1].

We now describe the syntax of $\text{PEPA}^{\infty}_{\text{ph}}$ terms. Let $a \in \mathcal{A}ct \cup \{\tau\}, S \subseteq \mathcal{A}ct$ and $p \in (0, 1)$. A valid term E of $\text{PEPA}^{\infty}_{\text{ph}}$ must be of the form

$$E ::= P \parallel_S Q(0)$$

where P is a PEPA[1] term, $Q(0)$ is a term belonging to the class $\text{QA}(i)$ of *infinite queue* terms, and $\parallel_S$ is the synchronization operator. We describe the classes PEPA and $\text{QA}(i)$ of terms in turn.

[1] Roughly speaking, the language $\text{PEPA}^{\infty}_{\text{ph}}$ subsumes PEPA, with the only differences arising through the interpretation of the synchronization operator.

The subset PEPA of terms is given by

$$P ::= (a, ph).P \mid P + P \mid P_p\oplus P \mid P \parallel_S P \mid P/S \mid C.$$

The meaning of the operators is based on those of PEPA [1], where we refer the reader for more detail. The prefix $(a, ph).P$, action choice $P + R$ and hiding P/S are as in PEPA, except we allow phase-type distributions instead of exponentials. The probabilistic choice operator, $P_p\oplus R$, is not included in PEPA, but can be modelled there by using passive activities. Thus, $P_p\oplus R$ represents a system which may behave either as the component P or as R, where the chosen activity is decided internally, with P chosen with probability p and R with probability $1 - p$. The synchronization operator $P \parallel_S R$ realises the same intuition as the PEPA cooperation $\underset{S}{\bowtie}$, i.e. the synchronization is at the rate of the slowest, except that we allow phase-type distributions, and so can take the maximum of the random variables directly. As is standard in PEPA, we assume the existence of a countable set of constants C. The meaning of a constant $C \stackrel{\text{def}}{=} P$ is given by the defining equation P.

Now we turn our attention to the *infinite queue terms* QA(i). Let i be an unbounded integer variable. Intuitively, each QA(i) term models the behaviour of an infinite queue parametrised by the variable i recording the number of customers, where $k^\star$ denotes the maximum number of customers that can be served at any time. Formally, the syntax of the sublanguage QA(i) is given by:

$$\begin{aligned}
Q(i) ::= \ & \texttt{if } i = 0 \texttt{ then} \\
& \quad Q_0 \\
& \texttt{else if } i = 1 \texttt{ then} \\
& \quad Q_1 \\
& \vdots \\
& \texttt{else if } i = j - 1 \texttt{ then} \\
& \quad Q_{j-1} \\
& \texttt{else} \\
& \quad (a, ph).Q(i + 1) \sqcap (b, ph').Q(i) \sqcap (c, ph'').Q(i - k^\star) \sqcap Q_j
\end{aligned}$$

where $\sqcap$ stands for either $+$ or ${}_p\oplus$, $j \geq k^\star \geq 1$, $Q_0, Q_1, \ldots, Q_{j-1} \in \Sigma^j_{Q(i)}$ and $Q_j \in \Sigma^{j-1}_{Q(i)}$. We use $\Sigma^k_{Q(i)}$, for any $k \in \mathbb{N}$, to denote the subset of $\text{PEPA}^\infty_{\text{ph}}$ terms restricted to $Q(i)$ terms only for $0 \leq i \leq k$, defined by the following syntax:

$$Q ::= (a, ph).Q(i) \mid Q + Q \mid Q \,{}_p\oplus Q$$

where $i \in \{0, \ldots, k\}$.

In the above, the terms $Q_0, Q_1, \ldots, Q_{i^\star-1}$ correspond to transitions within the initial portion and to the repetitive portion, Q_j corresponds to transitions from the repetitive to the initial portion, and for any $i \geq j$, the action a denotes a transition from level i to level $i+1$, b a transition within level i and c a transition from level i to level $i - k^\star$. Formally, each term $Q(i)$ in the sublanguage QA(i) denotes an infinite set of defining equations for constants

$$Q(0) \stackrel{\text{def}}{=} \cdots, \ \ldots, Q(i) \stackrel{\text{def}}{=} \cdots, \ \ldots$$

We note that the value of $i^\star$, the start of the repetitive portion, can be computed by reachability analysis on the subcomponents of $\mathrm{QA}(i)$ terms, namely $Q(0), \ldots, Q(j)$. In many cases the value of $i^\star$ can simply be read from the specification and often $i^\star = j$.

We now illustrate the above syntax by means of an example.

Example 1. Consider a queueing model with failure*, subject to the constraint only one customer is served at a time ($k^\star = 1$) and if the processor fails, all the customers in the system are lost. The state transition diagram for this queue is given in Figure 1, where the duration for arrival, service and failure are given by the phase-type distributions ph_1, ph_2 and ph_3 respectively.*

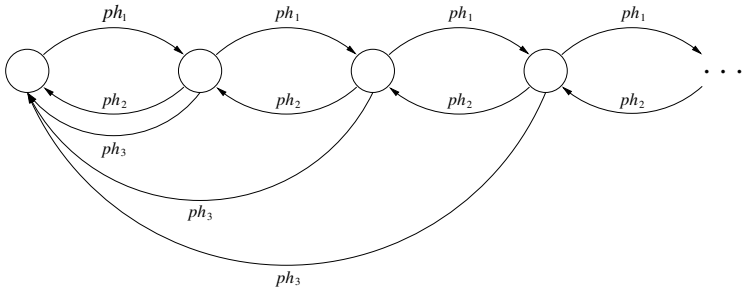

Fig. 1. State transition diagram for queueing model with failure

Using PEPA$^\infty_{\mathrm{ph}}$ this queue can be specified as follows:

$$
\begin{aligned}
\mathit{Queue}(i) \stackrel{\mathrm{def}}{=} \ & \texttt{if } i = 0 \texttt{ then} \\
& \quad (\mathit{arrive}, ph_1).\mathit{Queue}(1) \\
& \texttt{else} \\
& \quad (\mathit{arrive}, ph_1).\mathit{Queue}(i+1) + (\mathit{serve}, \top).\mathit{Queue}(i-1) \\
& \quad +(\mathit{fail}, \top).\mathit{Queue}(0) \\
\mathit{Server} \stackrel{\mathrm{def}}{=} \ & (\mathit{serve}, ph_2).\mathit{Server} + (\mathit{fail}, ph_3).\mathit{Server} \\
\mathit{System} \stackrel{\mathrm{def}}{=} \ & \mathit{Server} \underset{\{\mathit{serve},\, \mathit{fail}\}}{\|} \mathit{Queue}(0).
\end{aligned}
$$

By simple reachability analysis we find that $i^\star = 1$.

4.2 Operational Semantics

The operational semantics of PEPA$^\infty_{\mathrm{ph}}$ is given in Figure 2 below. Similarly to the semantics for PEPA [1], the transition rules of Figure 2 enable us to view a PEPA$^\infty_{\mathrm{ph}}$ component as a *multi-graph* whose nodes are components and arcs represent the possible transitions between them, as derived by the above rules.

We shall require the notion of the *derivative set* [1] of a node of the multi-graph corresponding to a PEPA$^\infty_{\mathrm{ph}}$ component, which we now introduce.

Definition 3. *The* derivative set *of a* PEPA$^\infty_{\mathrm{ph}}$ *component E, denoted $ds(E)$, is the smallest set of* PEPA$^\infty_{\mathrm{ph}}$ *components satisfying the following conditions:*

Prefix

$$\frac{}{(a,ph).P \xrightarrow{(a,ph)} P}$$

Action Choice

$$\frac{P \xrightarrow{(a,ph)} \tilde{P}}{P + R \xrightarrow{(a,ph)} \tilde{P}} \qquad \frac{R \xrightarrow{(a,ph)} \tilde{R}}{P + R \xrightarrow{(a,ph)} \tilde{R}}$$

Probabilistic Choice

$$\frac{P \xrightarrow{(a,ph)} \tilde{P}}{P_p\oplus R \xrightarrow{(a,p\cdot ph)} \tilde{P}} \qquad \frac{R \xrightarrow{(a,ph)} \tilde{R}}{P_p\oplus R \xrightarrow{(a,(1-p)\cdot ph)} \tilde{R}}$$

Synchronization

$$\frac{P \xrightarrow{(a,ph)} \tilde{P}}{P \parallel_S R \xrightarrow{(a,ph)} \tilde{P} \parallel_S R}(a \notin S) \qquad \frac{R \xrightarrow{(a,ph)} \tilde{R}}{P \parallel_S R \xrightarrow{(a,ph)} P \parallel_S \tilde{R}}(a \notin S)$$

$$\frac{P \xrightarrow{(a,ph_1)} \tilde{P} \;\; R \xrightarrow{(a,ph_2)} \tilde{R}}{P \parallel_S R \xrightarrow{(a,ph_{12})} \tilde{P} \parallel_S \tilde{R}}(a \in S) \;\; \textit{where} \;\; ph_{12} = max(ph_1, ph_2)$$

Hiding

$$\frac{P \xrightarrow{(a,ph)} \tilde{P}}{P/S \xrightarrow{(a,ph)} \tilde{P}/S}(a \notin S) \qquad \frac{P \xrightarrow{(a,ph)} \tilde{P}}{P/S \xrightarrow{(\tau,ph)} \tilde{P}/S}(a \in S)$$

Constant

$$\frac{P \xrightarrow{(a,ph)} \tilde{P}}{C \xrightarrow{(a,ph)} \tilde{P}}(C \stackrel{\text{def}}{=} P)$$

Fig. 2. Operational Semantics of $\text{PEPA}^{\infty}_{\text{ph}}$.

- *if* $E \stackrel{\text{def}}{=} E_0$ *then* $E_0 \in ds(E)$
- *if* $E_i \in ds(E)$ *and* $E_i \xrightarrow{(a,ph)} E_j$ *then* $E_j \in ds(E)$.

As in PEPA [1] we will require that $P \parallel_S Q(0)$ is *ergodic* and *complete* (there are no passive activities). We note that necessary conditions for this to hold include: P is cyclic, $S \subseteq \mathcal{A}ct(P) \cap \mathcal{A}ct(Q(i))$ (where for any component R, $\mathcal{A}ct(R)$ is the set of actions that appear in the multi-graph of R), and $Q(i^*) \in ds(Q(0))$.

5 Derivation of the Structure of the Generator Matrix

All mainstream Stochastic Process Algebra languages, i.e. EMPA [6], TIPP [5] and PEPA [1], advocate the use of the generator matrix $\mathbf{Q}$ of the underlying Markov process for calculating the performance measures of the system under study. Thus, a component C is converted into the generator matrix $\mathbf{Q}$, based on the rates of the activities as specified in the original component, and the steady-state distribution $\mathit{\Pi}(\cdot)$ of the system may be found by solving the (normalizing and global balance) equations:

$$\sum_{C_i \in ds(C)} \mathit{\Pi}(C_i) = 1 \text{ and } \mathit{\Pi}\mathbf{Q} = \mathbf{0}.$$

This vector of steady-state probabilities is used as input to a range of formulas which allow to evaluate the performance of the model under study. As we rely on the Matrix-Geometric Method, we will not construct the generator matrix $\mathbf{Q}$ for a finite approximation of the system, and instead derive for each $\text{PEPA}^{\infty}_{\text{ph}}$ component a family of (in general smaller) submatrices of the generator matrix $\mathbf{Q}$ as described in Section 3. Then we will apply the algorithm in Section 3 for computing the matrix $\mathbf{R}$, which will be used as input for performance characteristics.

For the remainder of this section we fix a $\text{PEPA}^{\infty}_{\text{ph}}$ component $E = P \parallel_S Q(0)$, and hence fix values for $k^\star$ (the number of customers that can be served at any one time) and $i^\star$ (the start of the repetitive portion).

The first step involves calculating the following sets of derivatives of $P \parallel_S Q(0)$. We say P' is *reachable* from P if $P' \in ds(P)$.

1. Construct the set *init* of components of the form $P' \parallel_S Q(i)$ that is reachable from $P \parallel_S Q(0)$, for some component P' and some $i \leq i^\star$ (the components in the *initial portion*).
2. Construct the set *rep* of components P' such that $P' \parallel_S Q(i^\star)$ is reachable from $P \parallel_S Q(0)$ (the components in the *repetitive portion*).

We note that, for any $i^\star + i$ in the repetitive portion where $i \geq 1$, the set of components P' such that $P' \parallel_S Q(i^\star + i)$ is reachable from $P \parallel_S Q(0)$ will equal *rep*.

Before we introduce our algorithm to construct the above sets we require the following definition. For any set of components $\mathcal{P}$ and $\sim \in \{<, =, >\}$ let

$$\mathit{Reach}(\mathcal{P}, \sim i^\star) \stackrel{\text{def}}{=} \{P' \parallel_S Q(i) \mid \exists\, R \in \mathcal{P} \text{ such that } R \xrightarrow{(a,ph)} P' \parallel_S Q(i) \text{ and } i \sim i^\star\}.$$

Intuitively, to find the sets *init* and *rep* we must perform reachability analysis of the multi-graph of $P \parallel Q(0)$. More precisely, using the repetitive structure of $\text{PEPA}^{\infty}_{\text{ph}}$ terms, we step through the multi-graph inductively, one transition step at a time, finding the reachable components of the required form and also keeping track of all the components we have visited until the sets *init* and *rep*

settle (in other words, the fixed point is reached). Furthermore, this process terminates after a finite number of steps since $P \parallel Q(0)$ is ergodic.

Algorithm 1
Input: A $\text{PEPA}^{\infty}_{\text{ph}}$ component $P \parallel_S Q(0)$.
Output: $init \subseteq ds(P \parallel_S Q(0))$ and $rep \subseteq ds(P)$.

Initialization: $init_0 = \{P \parallel_S Q(0)\}$, $rep_0 = \emptyset$ and $inf_0 = \emptyset$.
Repeat For $k \geq 0$, let

$$
\begin{aligned}
init_{k+1} &= Reach(init_k, < i^\star) \cup Reach(rep_k, < i^\star) \cup Reach(inf_k, < i^\star) \cup init_k \\
rep_{k+1} &= Reach(init_k, = i^\star) \cup Reach(rep_k, = i^\star) \cup Reach(inf_k, = i^\star) \cup rep_k \\
aux &= Reach(init_k, > i^\star) \cup Reach(rep_k, > i^\star) \cup Reach(inf_k, > i^\star) \cup inf_k \\
inf_{k+1} &= \{P' \parallel_S Q(i) \mid P' \parallel_S Q(i) \in aux \text{ and } P' \parallel_S Q(i^\star) \notin rep_{k+1}\}.
\end{aligned}
$$

Until $init_{k+1} = init_k$, $rep_{k+1} = rep_k$ and $inf_{k+1} = \emptyset$.
Return: $init = init_k$ and $rep = \{P' \mid P' \parallel_S Q(i^\star) \in rep_k\}$.

We illustrate the working of Algorithm 1 using Example 1, recalling that $k^\star = i^\star = 1$. Let $S = \{serve, fail\}$ then the algorithm computes the sets:

$$
\begin{aligned}
init_0 &= \{Server \parallel_S Queue(0)\} \\
rep_0 &= \emptyset \\
inf_0 &= \emptyset \\
init_1 &= \{Server \parallel_S Queue(0)\} \\
rep_1 &= \{Server \parallel_S Queue(1)\} \\
inf_1 &= \emptyset \\
init_2 &= \{Server \parallel_S Queue(0)\} \\
rep_2 &= \{Server \parallel_S Queue(1)\} \\
inf_2 &= \emptyset
\end{aligned}
$$

and hence $init = \{Server \parallel_S Queue(0)\}$ and $rep = \{Server\}$.

The sets of derivatives calculated by Algorithm 1 serve as input to Algorithm 2 given below. The aim of this algorithm is to compute a family of submatrices (denoted with the help of apostrophe, e.g. $\mathbf{B'_{1,0}}$). These are not directly the submatrices of the generator matrix $\mathbf{Q}$ as described in Section 3, but instead *descriptors* for these submatrices, in the sense that a single location in such a matrix is used as a place holder for the transition rate, which can be a phase-type distribution of any size. The actual phase-type distributions can be "plugged into" the descriptor submatrices, taking care to match the sizes correctly with the help of Kronecker product $\otimes$, thus giving rise to submatrices of the generator matrix $\mathbf{Q}$.

Recall that we take the minimum of the durations (phase-type distributions) arising from action choice[2].

[2] If the matrices corresponding to the phases are of the same size, the minimum reduces to summation of the matrices.

Algorithm 2
Input: Two finite sets of $\text{PEPA}^{\infty}_{\text{ph}}$ and PEPA components *init* and *rep*.
Output: Matrices $\mathbf{B'_{0,0}}$, $\mathbf{B'_{0,1}}$, $\mathbf{B'_{k,0}}$ for $\mathbf{k} \geq \mathbf{1}$, $\mathbf{A'_0}$, $\mathbf{A'_1}$ and $\mathbf{A'_{k^\star+1}}$.

Step 1 Enumerate the sets *init* and *rep* such that $init = \{E^0_1, \ldots, E^0_{m_0}\}$ and $rep = \{P_1, \ldots, P_m\}$.

Step 2 Construct the $m_0 \times m_0$ matrices $\mathbf{B'_{0,0}}$ and $\mathbf{B'_{1,0}}$, $m_0 \times m$ matrices $\mathbf{B'_{k,0}}$ for $\mathbf{k} \geq \mathbf{1}$, and $m \times m$ matrices $\mathbf{A'_0}$, $\mathbf{A'_1}$ and $\mathbf{A'_{k^\star+1}}$ as follows, where below we let a ranges over the set of actions $\mathcal{A}ct$ and $\min\{\emptyset\} = 0$:

$$\begin{aligned}
\mathbf{B'_{0,0}}(i,j) &= \min\{ph \mid E^0_i \xrightarrow{(a,ph)} E^0_j\}\\
\mathbf{B'_{0,1}}(i,j) &= \min\{ph \mid E^0_i \xrightarrow{(a,ph)} P_j \parallel_S Q(i^\star)\}\\
\mathbf{B'_{k,0}}(i,j) &= \min\{ph \mid P_i \parallel_S Q(k+i^\star) \xrightarrow{(a,ph)} E^0_j\}\\
\mathbf{A'_0}(i,j) &= \min\{ph \mid P_i \parallel_S Q(i^\star) \xrightarrow{(a,ph)} P_j \parallel_S Q(i^\star+1)\}\\
\mathbf{A'_1}(i,j) &= \min\{ph \mid P_i \parallel_S Q(i^\star) \xrightarrow{(a,ph)} P_j \parallel_S Q(i^\star)\}\\
\mathbf{A'_{k^\star+1}}(i,j) &= \min\{ph \mid P_i \parallel_S Q(i^\star+k^\star) \xrightarrow{(a,ph)} P_j \parallel_S Q(i^\star)\}.
\end{aligned}$$

If we consider Example 1 again, recall that $k^\star = i^\star = 1$ and Algorithm 1 gave us the sets $init = \{Server \parallel_S Queue(0)\}$ and $rep = \{Server\}$. Then, the matrices (all of size 1×1) generated by Algorithm 2 are:

$$\begin{aligned}
\mathbf{B'_{0,0}}(1,1) &= \min\{ph \mid Server \parallel_S Queue(0) \xrightarrow{(a,ph)} Server \parallel_S Queue(0)\}\\
&= 0\\
\mathbf{B'_{0,1}}(1,1) &= \min\{ph \mid Server \parallel_S Queue(0) \xrightarrow{(a,ph)} Server \parallel_S Queue(1)\}\\
&= ph_1\\
\mathbf{B'_{1,0}}(1,1) &= \min\{ph \mid Server \parallel_S Queue(1) \xrightarrow{(a,ph)} Server \parallel_S Queue(0)\}\\
&= \min\{ph_2, ph_3\}\\
\mathbf{B'_{k,0}}(1,1) &= \min\{ph \mid Server \parallel_S Queue(k+1) \xrightarrow{(a,ph)} Server \parallel_S Queue(0)\}\\
&= ph_3\\
\mathbf{A'_0}(1,1) &= \min\{ph \mid Server \parallel_S Queue(1) \xrightarrow{(a,ph)} Server \parallel_S Queue(2)\}\\
&= ph_1\\
\mathbf{A'_1}(1,1) &= \min\{ph \mid Server \parallel_S Queue(1) \xrightarrow{(a,ph)} Server \parallel_S Queue(1)\}\\
&= 0\\
\mathbf{A'_2}(1,1) &= \min\{ph \mid Server \parallel_S Queue(2) \xrightarrow{(a,ph)} Server \parallel_S Queue(1)\}\\
&= ph_2.
\end{aligned}$$

for $\mathbf{k} \geq \mathbf{2}$.

As mentioned above, the matrices calculated by Algorithm 2 are only descriptors of the submatrices of the generator matrix $\mathbf{Q}$. They can be used to generate these submatrices by matching the sizes of the phase distributions. Suppose $ph_1 = (\underline{\alpha}_1, \mathbf{T_1})$, $ph_2 = (\underline{\alpha}_2, \mathbf{T_2})$, $ph_3 = (\underline{\alpha}_3, \mathbf{T_3})$ and $ph_4 = \min\{ph_2, ph_3\} = (\underline{\alpha}_4, \mathbf{T_4})$, then we calculate the submatrices of the generator matrix for Example 1 by performing the following steps[3].

1. Calculate $\mathbf{A_0}$, $\mathbf{A_1}$ and $\mathbf{A_2}$ according to the following equations:

$$\begin{aligned}
\mathbf{A_0} &= \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I}_{\mathbf{T_4}} \\
\mathbf{A_1} &= (\mathbf{T_1} \otimes \mathbf{I}_{\mathbf{T_4}}) + (\mathbf{I}_{\mathbf{T_1}} \otimes \mathbf{I}_{\mathbf{T_3}} \otimes \mathbf{T_2}) + (\mathbf{I}_{\mathbf{T_1}} \otimes \mathbf{I}_{\mathbf{T_2}} \otimes \mathbf{T_3}) \\
\mathbf{A_2} &= \mathbf{I}_{\mathbf{T_1}} \otimes \mathbf{I}_{\mathbf{T_3}} \otimes \underline{T}_2^0 \underline{\alpha}_2.
\end{aligned}$$

2. Calculate $\mathbf{B_{0,0}}$, $\mathbf{B_{0,1}}$, $\mathbf{B_{1,0}}$ and $\mathbf{B_{k,0}}$ for $\mathbf{k \geq 2}$. First, since the matrices $\mathbf{B_{k,0}}$ and $\mathbf{B_{1,0}}$ need to be of equal size, we construct a *new* phase-type distribution ph_5, of the same size as ph_4 and equivalent to ph_3, by multiplying $\mathbf{T_3}$ by the identity matrix of $\mathbf{T_2}$ and calculating $\underline{T}_5^0$. We now construct the matrices according to the following equations:

$$\begin{aligned}
\mathbf{B_{0,0}} &= \mathbf{T_1} \\
\mathbf{B_{0,1}} &= \underline{T}_1^0 \underline{\alpha}_1 \otimes \underline{\alpha}_4 \\
\mathbf{B_{1,0}} &= \mathbf{I}_{\mathbf{T_1}} \otimes \underline{T}_4^0 \\
\mathbf{B_{k,0}} &= \mathbf{I}_{\mathbf{T_1}} \otimes \underline{T}_5^0.
\end{aligned}$$

6 Examples

We now illustrate the use of PEPA$^{\infty}_{\text{ph}}$ and our method of analysis by means of modelling representative queueing systems. For simplicity, we have restricted choice to action choice (+). However, the case of probabilistic choice (or a mixture of the two types of choice) can be handled similarly, except that the resulting phase-type distributions in the matrices constructed by Algorithm 2 will have to be amended according to the transition rules.

6.1 Quasi-Birth-Death Queue

Consider the $ph/ph/1$ queueing model, which is an infinite queue of arrivals with arrival and service durations given by the phase-type distributions ph_1 and ph_2 respectively; it is a Quasi-Birth-Death process. Its PEPA$^{\infty}_{\text{ph}}$ description is as follows:

$$\begin{aligned}
Queue(i) &\stackrel{\text{def}}{=} \texttt{if } i = 0 \texttt{ then} \\
&\qquad (arrive, ph_1).Queue(1) \\
&\quad \texttt{else} \\
&\qquad (arrive, ph_1).Queue(i+1) + (serve, \top).Queue(i-1) \\
Server &\stackrel{\text{def}}{=} (serve, ph_2).Server \\
System &\stackrel{\text{def}}{=} Queue(0) \parallel_{serve} Server.
\end{aligned}$$

[3] Recall that by definition of min: $\underline{\alpha}_4 = \underline{\alpha}_2 \otimes \underline{\alpha}_3$ and $\mathbf{T_4} = (\mathbf{T_2} \otimes \mathbf{I}_{\mathbf{T_3}}) + (\mathbf{I}_{\mathbf{T_2}} \otimes \mathbf{T_3})$.

In this case we have $i^\star = k^\star = 1$. Now, using Algorithm 1 we find that $init = \{Server \parallel_{\{serve\}} Queue(0)\}$ and $rep = \{Server\}$. Next we use Algorithm 2 to construct the family of descriptor matrices which determine the submatrices of the actual generator matrix $\mathbf{Q}$ and find that the only non-zero matrices are as shown below:

$$\mathbf{B'_{0,1}} = \mathbf{A'_0} = (ph_1) \quad \text{and} \quad \mathbf{B'_{1,0}} = \mathbf{A'_2} = (ph_2).$$

Now supposing $ph_1 = (\underline{\alpha}_1, \mathbf{T_1})$ and $ph_2 = (\underline{\alpha}_2, \mathbf{T_2})$ we calculate the the submatrices of the generator matrix $\mathbf{Q}$ can be calculated as follows:

1. Calculate $\mathbf{A_0}$, $\mathbf{A_1}$ and $\mathbf{A_2}$ according to the following equations:

$$\begin{aligned}
\mathbf{A_0} &= \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I_{T_2}} \\
\mathbf{A_1} &= (\mathbf{T_1} \otimes \mathbf{I_{T_2}}) + (\mathbf{I_{T_1}} \otimes \mathbf{T_2}) \\
\mathbf{A_2} &= \mathbf{I_{T_1}} \otimes \underline{T}_2^0 \underline{\alpha}_2.
\end{aligned}$$

2. Calculate $\mathbf{B_{0,0}}$, $\mathbf{B_{0,1}}$ and $\mathbf{B_{1,0}}$ according to the following equations:

$$\begin{aligned}
\mathbf{B_{0,0}} &= \mathbf{T_1} \\
\mathbf{B_{0,1}} &= \underline{T}_1^0 \underline{\alpha}_1 \otimes \underline{\alpha}_2 \\
\mathbf{B_{1,0}} &= \mathbf{I_{T_1}} \otimes \underline{T}_2^0.
\end{aligned}$$

6.2 Bulk Service Queue

A *bulk service* queue, in which the server can serve up to k customers at a time, with duration given by the phase-type distribution ph_2 and arrival duration given by ph_1, is described by:

$$\begin{aligned}
Queue(i) \stackrel{\text{def}}{=} & \texttt{ if } i = 0 \texttt{ then} \\
& \quad (Queue, ph_1).Queue(1) \\
& \texttt{else if } i = 1 \texttt{ then} \\
& \quad (Queue, ph_1).Queue(2) + (serve, \top).Queue(0) \\
& \quad \vdots \\
& \texttt{else if } i = k^\star \texttt{ then} \\
& \quad (Queue, ph_1).Queue(k^\star + 1) + (serve, \top).Queue(0) \\
& \texttt{else} \\
& \quad (Queue, ph_1).Queue(i + 1) + (serve, \top).Queue(i - k^\star) \\
Server \stackrel{\text{def}}{=} & (serve, ph_2).Server \\
System \stackrel{\text{def}}{=} & Server \parallel_{\{serve\}} Queue(0).
\end{aligned}$$

In this case $i^\star = k^\star = k$ where k is the number of customers served at a time. Using Algorithm 1 and Algorithm 2 we have:

$$\begin{aligned}
&init = \{Server \parallel_{\{serve\}} Queue(0), \ \ldots, \ Server \parallel_{\{serve\}} Queue(k^\star - 1)\}, \\
&rep = \{Server\}
\end{aligned}$$

and the non-zero matrices:

$$\mathbf{B'_{0,0}} = \begin{pmatrix} 0 & ph_1 & 0 & \dots & 0 \\ ph_2 & 0 & ph_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ ph_2 & 0 & 0 & \dots & ph_1 \\ ph_2 & 0 & 0 & \dots & 0 \end{pmatrix}, \quad \mathbf{B'_{0,1}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ ph_1 \end{pmatrix}$$

$$\begin{aligned} \mathbf{B'_{1,0}} &= (ph_2\ 0\ 0\ \dots\ 0), \\ \mathbf{B'_{2,0}} &= (0\ ph_2\ 0\ \dots\ 0), \\ \vdots\ \vdots & \qquad\qquad \vdots \\ \mathbf{B'_{k^\star,0}} &= (0\ 0\ 0\ \dots\ ph_2), \end{aligned}$$

$$\mathbf{A'_0} = (ph_1) \text{ and } \mathbf{A'_{k^\star+1}} = (ph_2).$$

Now supposing $ph_1 = (\underline{\alpha}_1, \mathbf{T_1})$ and $ph_2 = (\underline{\alpha}_2, \mathbf{T_2})$, the submatrices of the generator matrix $\mathbf{Q}$ can be derived by the following steps.

1. Calculate $\mathbf{A_0}$, $\mathbf{A_1}$ and $\mathbf{A_{k^\star+1}}$ according to the equations:

$$\begin{aligned} \mathbf{A_0} &= \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I_{T_2}} \\ \mathbf{A_1} &= (\mathbf{T_1} \otimes \mathbf{I_{T_2}}) + (\mathbf{I_{T_1}} \otimes \mathbf{T_2}) \\ \mathbf{A_{k^\star+1}} &= \mathbf{I_{T_1}} \otimes \underline{T}_2^0 \underline{\alpha}_2. \end{aligned}$$

2. Calculate $\mathbf{B_{0,0}}$, $\mathbf{B_{0,1}}$, $\mathbf{B_{1,0}}$ and $\mathbf{B_{k,0}}$ for $\mathbf{2} \leq \mathbf{k} \leq \mathbf{k^\star}$ according to the following equations:

$$\mathbf{B'_{0,0}} = \begin{pmatrix} \mathbf{T_1} & \underline{T}_1^0 \underline{\alpha}_1 \otimes \underline{\alpha}_2 & 0 & \dots & 0 \\ \mathbf{I_{T_1}} \otimes \underline{T}_2^0 & \mathbf{T_1} \otimes \mathbf{I_{T_2}} + \mathbf{I_{T_1}} \otimes \mathbf{T_2} & \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I_{T_2}} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I_{T_1}} \otimes \underline{T}_2^0 & 0 & 0 & \dots & \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I_{T_2}} \\ \mathbf{I_{T_1}} \otimes \underline{T}_2^0 & 0 & 0 & \dots & \mathbf{T_1} \otimes \mathbf{I_{T_2}} + \mathbf{I_{T_1}} \otimes \mathbf{T_2} \end{pmatrix}$$

$$\mathbf{B_{0,1}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \underline{T}_1^0 \underline{\alpha}_1 \otimes \mathbf{I_{T_2}} \end{pmatrix} \qquad \begin{aligned} \mathbf{B_{1,0}} &= (\mathbf{I_{T_1}} \otimes \underline{T}_2^0\ 0\ 0\ \dots\ 0) \\ \mathbf{B_{2,0}} &= (0\ \mathbf{I_{T_1}} \otimes \underline{T}_2^0\ 0\ \dots\ 0) \\ \vdots\ \vdots & \qquad\qquad \vdots \\ \mathbf{B_{k^\star,0}} &= (0\ 0\ 0\ \dots\ \mathbf{I_{T_1}} \otimes \underline{T}_2^0). \end{aligned}$$

6.3 Multimedia Stream

This model is taken from [19], where it was stated that it would be desirable to analyse it using an unbounded buffer and Erlang distributed timeouts. The basic multi-media stream consists of the following components: a source process which generates a continuous sequence of packets, a medium (the channel) which supports asynchronous communication, and the sink which displays the packets.

Furthermore, the timer monitors delays between displays. In [19] this stream is both specified and analyzed using PEPA, and hence only a finite approximation of the model is considered. Here we use $\text{PEPA}^{\infty}_{\text{ph}}$, and hence can represent the channel as an unbounded buffer of packets, as given below:

$$
\begin{aligned}
\mathit{Channel}(i) &\stackrel{\text{def}}{=} \texttt{if } i = 0 \texttt{ then} \\
&\qquad (\mathit{transmit}, \top).\mathit{Channel}(1) \\
&\quad \texttt{else} \\
&\qquad (\mathit{transmit}, \top).\mathit{Channel}(i+1) + (\mathit{receive}, ph_1).\mathit{Channel}(i-1) \\
\mathit{Source} &\stackrel{\text{def}}{=} (\mathit{transmit}, ph_2).\mathit{Source} \\
\mathit{Sink}_0 &\stackrel{\text{def}}{=} (\mathit{receive}, \top).\mathit{Sink}_1 \\
\mathit{Sink}_1 &\stackrel{\text{def}}{=} (\mathit{receive}, \top).\mathit{Sink}_2 + (\mathit{display}, ph_3).\mathit{Sink}_0 + (\mathit{timeout}, \top).\mathit{Sink}_0 \\
\mathit{Sink}_2 &\stackrel{\text{def}}{=} (\mathit{receive}, \top).\mathit{Sink}_3 + (\mathit{display}, ph_3).\mathit{Sink}_1 + (\mathit{timeout}, \top).\mathit{Sink}_2 \\
\mathit{Sink}_3 &\stackrel{\text{def}}{=} (\mathit{display}, ph_3).\mathit{Sink}_2 + (\mathit{timeout}, \top).\mathit{Sink}_1 \\
\mathit{Timer} &\stackrel{\text{def}}{=} (\mathit{tick}, ph_4).\mathit{Tick} \\
\mathit{Tick} &\stackrel{\text{def}}{=} (\mathit{timeout}, ph_5).\mathit{Timeout} \\
\mathit{System} &\stackrel{\text{def}}{=} (\mathit{Source} \parallel \mathit{Sink}_0 \parallel_{\{\mathit{timeout}\}} \mathit{Timer}) \parallel_{\{\mathit{transmit},\mathit{receive}\}} \mathit{Channel}(0).
\end{aligned}
$$

For this example $i^\star = k^\star = 1$ and by Algorithm 1 we have sets of derivatives:

$$
\begin{aligned}
\mathit{init} = \{&\mathit{Source} \parallel \mathit{Sink}_0 \parallel \mathit{Timer} \parallel \mathit{Channel}(0),\ \mathit{Source} \parallel \mathit{Sink}_0 \parallel \mathit{Tick} \parallel \mathit{Channel}(0), \\
&\mathit{Source} \parallel \mathit{Sink}_1 \parallel \mathit{Timer} \parallel \mathit{Channel}(0),\ \mathit{Source} \parallel \mathit{Sink}_1 \parallel \mathit{Tick} \parallel \mathit{Channel}(0), \\
&\mathit{Source} \parallel \mathit{Sink}_2 \parallel \mathit{Timer} \parallel \mathit{Channel}(0),\ \mathit{Source} \parallel \mathit{Sink}_2 \parallel \mathit{Tick} \parallel \mathit{Channel}(0), \\
&\mathit{Source} \parallel \mathit{Sink}_3 \parallel \mathit{Timer} \parallel \mathit{Channel}(0),\ \mathit{Source} \parallel \mathit{Sink}_3 \parallel \mathit{Tick} \parallel \mathit{Channel}(0)\}
\end{aligned}
$$

and

$$
\begin{aligned}
\mathit{rep} = \{&\mathit{Source} \parallel \mathit{Sink}_0 \parallel \mathit{Timer},\ \mathit{Source} \parallel \mathit{Sink}_0 \parallel \mathit{Tick}, \\
&\mathit{Source} \parallel \mathit{Sink}_1 \parallel \mathit{Timer},\ \mathit{Source} \parallel \mathit{Sink}_1 \parallel \mathit{Tick}, \\
&\mathit{Source} \parallel \mathit{Sink}_2 \parallel \mathit{Timer},\ \mathit{Source} \parallel \mathit{Sink}_2 \parallel \mathit{Tick}, \\
&\mathit{Source} \parallel \mathit{Sink}_3 \parallel \mathit{Timer},\ \mathit{Source} \parallel \mathit{Sink}_3 \parallel \mathit{Tick}\}.
\end{aligned}
$$

Then using Algorithm 2 we generate the matrices:

$$
\mathbf{B}'_{\mathbf{0},\mathbf{1}} = \mathbf{A}'_{\mathbf{0}} = ph_2 \cdot \mathbf{I},
$$

$$
\mathbf{B}'_{\mathbf{0},\mathbf{0}} = \mathbf{A}'_{\mathbf{1}} = \begin{pmatrix}
0 & ph_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
ph_3 & 0 & 0 & ph_4 & 0 & 0 & 0 & 0 \\
ph_5 & ph_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & ph_3 & 0 & 0 & ph_4 & 0 & 0 \\
0 & 0 & ph_5 & ph_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & ph_3 & 0 & 0 & ph_4 \\
0 & 0 & 0 & 0 & ph_5 & ph_3 & 0 & 0
\end{pmatrix}
$$

$$\mathbf{B'_{10}} = \mathbf{A'_2} = \begin{pmatrix} 0 & 0 & ph_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & ph_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ph_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ph_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & ph_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & ph_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Similarly to the other examples we can then calculate the actual submatrices of the generator matrix.

6.4 Calculating Performance Measures

The performance characteristics for the examples discussed in Subsections 6.1–6.3 can be computed as follows:

1. Calculate $\underline{\pi}_0$ and $\underline{\pi}_1$ using (3) (Subsection 3.2).
2. Calculate $\mathbf{R}$ using the iteration method or the Latouche Ramaswami algorithm [17] in the case of QBD queues (Subsection 3.3).
3. Calculate the required performance measures of the model (Subsection 3.4).

As demonstrated by examples, the proposed method enables us to consider the effect of using different phase-type distributions in the same model, without having to repeatedly perform reachability analysis of the underlying multi-graph. For example, if the queue is $M/E_2/1$ or $M/H_2(k)/1$ or $M/M/1$, we only redefine the values of ph_1 and ph_2 and calculate the submatrices of the generator matrix directly (that is, we need only apply Algorithm 1 and Algorithm 2 once).

We have compared PEPA (using the PEPA Workbench [20] and MATLAB solver) against our MATLAB implementation for calculating the steady state probabilities through MGM, and obtained the following results. In all cases the results of PEPA and $\text{PEPA}^{\infty}_{\text{ph}}$ agree up to three decimal places. For each case we have computed $\mathbf{R}$ by the iteration method (with $\varepsilon = 2.4 \times 10^{-5}$). For QBD models we have also used the Latouche and Ramaswami (LR) algorithm, with the same accuracy, but this method cannot be applied to more general queues.

Quasi-Birth-Death Queue (Subsection 6.1) We have considered the following phase-type distributions:

– exponential arrivals and hyper-exponential services with two phases.
In PEPA, approximating the infinite queue with 20 customers, the size of the generator matrix is 43×43 and the number of the floating point operations equals 10699. On the other hand, when the MGM method is used, the largest matrix is 3×3, 26 iterations are need to calculate $\mathbf{R}$ and the number of floating point operations is 2882, which reduces to 2470 and 3 iterations when the LR algorithm is used.

– exponential arrivals and hypo-exponential services.
  The number of floating point operations in PEPA (with 50 customers) equals 18925 and the size of the generator matrix is 103×103. Using the MGM method the largest matrix is 3×3, 2310 floating point operations are performed and 27 iterations are required to calculate $\mathbf{R}$, which reduces to 1990 floating point operations and 4 iterations using LR.

The queueing process with failure (Example 1) We assume that if the processor fails all customers are lost, arrivals are exponential, and services and failures are hypo-exponential. In PEPA, approximating with 50 customers, the Markov generator matrix is 103×103 and the number of floating point operations is 18996. For the MGM method, the largest matrix is 5×5 and 13988 floating point operations and 24 iterations are needed.

The bulk service queue (Subsection 6.2) We work with exponential rate of arrival and exponential service rate. In PEPA (approximating with 50 customers) the number of floating point operations equals 13707 flops and the generator matrix is of size 52×52. For the MGM method 437 floating point operations and 4 iterations are required, and the largest matrix is 3×3.

Multi-media stream (Subsection 6.3) We use the distributions proposed in [19]: all durations are exponential except timeout which is Erlang. In PEPA (approximating with 50/100 channels) the generator matrix has size 1040×1040, respectively 2040×2040, and 2506175/25909274 floating point operations are required to calculate the steady state probabilities. Using the MGM method, the largest matrix is of size 40×40, 973322 floating point operations are required and 17 iterations to calculate $\mathbf{R}$.

7 Conclusion

Though performance evaluation of many systems can be achieved with the help of languages or models that only allow for the activity durations to be represented as *exponential distributions*, this restriction is not always realistic. In this paper we have introduced a new SPA language $\text{PEPA}^{\infty}_{\text{ph}}$ based on PEPA [1], which can be used to analyse performance of queueing models subject to certain restrictions on the structure and with the property that the duration of activities is modelled by *phase-type distributions*.

We use the $\text{PEPA}^{\infty}_{\text{ph}}$ language to describe representative examples of systems with potentially infinitely many customers. We formulate algorithms which, for a given valid $\text{PEPA}^{\infty}_{\text{ph}}$ component, calculate descriptors for submatrices of the generator matrix $\mathbf{Q}$ of the underlying Markov process. Our method permits the evaluation of the performance characteristics of the model by "plugging in" different phase-type distributions for each activity, without having to re-compute the submatrices of the generator matrix. The state space explosion is avoided by

using the Matrix-Geometric Method. First experimental results comparing the performance of the PEPA workbench with our approach are very encouraging.

Our method has so far been developed for a restricted class of queueing systems, as exemplified by the model studied. However, we note that there is a similarity between the construction in all the examples. We hope to be able to develop our framework further, so that it can be applied to more general classes of queueing models.

Finally, we note that an extension of TIPP similar to ours has been presented in [14], except that the spectral expansion method [15] is used instead of MGM. It would be interesting to compare the two methods as it is mentioned in [18] that the spectral expansion method is more efficient than MGM.

Acknowledgements

The authors are grateful to Jane Hillston, Joost-Pieter Katoen, Ulrich Herzog, Markus Siegle and the reviewers for comments on an earlier version of this work.

The last two authors are supported in part by EPSRC grant GR/M04617, and all are members of the ARC project 1031 *Stochastic Modelling and Verification* funded by the British Council and DAAD.

References

1. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
2. M. F. Neuts. *Matrix Geometric Solutions for Stochastic Models.* Johns Hopkins University Press, Baltimore, Maryland, 1981.
3. R. Nelson. *Probability, Stochastic Processes, and Queueing Theory.* Springer-Verlag, 1995.
4. W.J. Stewart. *Introduction to the Numerical Solutions of Markov Chains.* Princeton University Press, 1994.
5. M. Rettelbach and M. Siegle. Compositional minimal semantics for the stochastic process algebra TIPP. Technical Report 4, University of Erlangen, 1994.
6. M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.
7. M. Bravetti, M. Bernardo, and R. Gorrieri. Towards performance evaluation with general distributions in process algebras. In D. Sangiorgi and R. de Simone, editors, *Proc. CONCUR'98*, volume 1466 of *Lecture Notes in Computer Science*, pages 405–422. Springer, 1998.
8. H. Hermanns. *Interactive Markov chains.* PhD thesis, University of Erlangen, 1998.
9. P.R. D'Argenio, J-P. Katoen, and E. Brinksma. General purpose discrete event simulation using spades. In C. Priami, editor, *PAPM'98*, pages 85–102, 1998.
10. P.G. Harrison and B. Strulo. Stochastic process algebra for discrete event simulation. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parrallel Systems*, pages 18–34. Springer, 1995.

11. H. Hermanns and J Katoen. Automated compositional Markov chain generation for a plain-old telephone system. *Science of Computer Programming*, to appear.
12. B. R. Haverkort. Matrix-geometric solution of infinite stochastic Petri nets. In *IEEE International Computer Performance and Dependability Symposium*, pages 72–81. IEEE Computer Society Press, 1995.
13. H. Leemans. Queue lengths and waiting times in the two-class two-server queue with nonpreemptive heterogeneous priority structures. In R. Pooley and N. Thomas, editors, *Fourteenth UK Computer and Telecommunications Performance Engineering Workshop*, pages 150–164. The University of Edinburgh, 1998.
14. I. Mitrani, A. Ost, and M. Rettelbach. TIPP and the spectral expansion method. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, ESPRIT Basic Research Series, pages 99–113. Springer, 1995.
15. I. Mitrani and R. Chakka. Spectral expansion solution of a class of Markov models: applications and comparison with the matrix-geometric method. *Performance Evaluation*, 23:241–260, 1995.
16. M. Neuts. Two further closure properties of Ph-distributions. *Asia-Pacific Journal of Operational Research*, 9(1):77–85, 1992.
17. G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi birth and death processes. *Journal of Applied Probability*, 30:650–674, 1993.
18. B.R. Haverkort and A. Ost. Steady-state analysis of infinite stochastic Petri nets: comparing the spectral expansion and the matrix-geometric method. In *Proc. 7th Int. Workshop on Petri Nets and Performance Models*, pages 36–45. IEEE Computer Society Press, 1997.
19. H. Bowman, J.W. Bryans, and J. Derrick. Analysis of a multimedia stream using stochastic process algebra. In C. Priami, editor, *Proc. PAPM'98*, pages 51–69, 1998.
20. S. Gilmore and J. Hillston. The PEPA Workbench: a tool to support a process algebra-based approach to performance modelling. In *Proc. 7th Int. Conference on Modelling Techniques and Tools for Computer Performance Analysis*, volume 794 of *Lecture Notes in Computer Science*, pages 353–368. Springer, 1994.