An Efficient Kronecker Representation for PEPA Models *

Jane Hillston[1] and Leïla Kloul[2]

[1] LFCS, University of Edinburgh, Kings Buildings, Edinburgh EH9 3JZ, Scotland
jeh@dcs.ed.ac.uk
[2] PRiSM, Université de Versailles, 45, Av. des Etats-Unis, 78000 Versailles, France
kle@prism.uvsq.fr

Abstract. In this paper we present a representation of the Markov process underlying a PEPA model in terms of a Kronecker product of terms. Whilst this representation is similar to previous representations of Stochastic Automata Networks and Stochastic Petri Nets, it has novel features, arising from the definition of the PEPA models. In particular, capturing the correct timing behaviour of cooperating PEPA activities relies on functional dependencies.

1 Introduction

Performance investigation of modern computer and communication systems requires the development of relevant and efficient modelling techniques. The rich synchronisation constraints and the size of these systems lead to complex models with exponential growth of the number of states. Traditional performance models, based on queueing networks, cannot readily capture these constraints; thus several new performance modelling techniques have been developed, e.g. Stochastic Petri Nets (SPN), Stochastic Automata Networks (SAN) and Stochastic Process Algebras (SPA).

Petri nets were designed to represent synchronisation constraints within concurrent systems and protocols; SPN associate random variables with timed transitions within the net [16, 1, 20]. However, although the graphical representation of Petri Nets presents the dynamic behaviour of the model, it provides little insight into the structure of the system being modelled.

SAN and SPA provide mechanisms which allow the increasing complexity of synchronisation constraints to be captured whilst retaining the compositional structure of the system explicitly within the model. For many modern systems, being able to construct a model from components or elements, reflecting the system's composition, greatly aids handling the complexity of the model construction task. As with all state-based modelling formalisms, such models are prone to state space explosion. However, both formalisms incorporate techniques for overcoming this problem.

* This work is supported by CNRS/RS project (UIIVV 78171) and EPSRC project COMPA (G/L10215).

The SAN formalism, developed by Plateau [18], models complex systems with interacting components such as parallel systems. To tackle the state space explosion problem, Plateau [17] has proved that the generator matrix of the Markov process underlying a SAN model can be analytically represented using Kronecker algebra. Moreover the solution of the model can be achieved via this tensor expression of submatrices—the complete matrix does not need to be generated.

SPA are extensions of classical process algebras such as CCS and CSP, analogous to SPN in the sense that random variables are associated with timed actions in the model. In this paper we consider the Markovian process algebra, Performance Evaluation Process Algebra (PEPA), introduced by Hillston in 1994 [14]. In PEPA, a system is described as an interaction of components which, either singly or multiply, engage in activities. The components represent the active parts within the system and the activities the actions of those parts.

Various techniques for solving large models have been developed for PEPA but these have focused on aggregation or decomposition techniques, which use the process algebra structure of the model to guide manipulations of the underlying Markov process. In this paper we show that a PEPA model can also be represented analytically using Kronecker algebra and solved without constructing the complete generator matrix. Correct representation of the features of the PEPA model, in particular the synchronisation behaviour, relies on the functional dependencies introduced in PEPA formalism in [15]. Just as for SAN models, we show that the translation from the model to the compact representation is automatic.

This paper is structured as follows. In Sect. 2, we present the PEPA language. A small example illustrates the use of this modelling technique. Section 3 is dedicated to the functional depencies in PEPA. In Sect. 4, we show how to represent the underlying Markov process of a PEPA model using the tensor algebra. An application example is given, followed by the proof of the validity of this analytical representation. Section 5 is dedicated to related work. Finally, we conclude with some remarks and future work.

2 PEPA

The basic elements of PEPA[14] are *components* and *activities*, corresponding to *states* and *transitions* in the underlying Markov process. Each activity has an *action type* and τ denotes the distinguished type representing private or unseen action. The duration of each activity is represented by the parameter of the associated exponential distribution: the *activity rate* of the activity. The rate may be any positive real number, or the distinguished symbol $\top$ (read as *unspecified*). Thus each activity, a, is a pair (α, r) where α is the type and r is the rate. We let $\mathcal{C}$ denote the countable set of components and $\mathcal{A}$ denote the countable set of all possible action types. We denote by $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$, the set of activities, where $\mathbb{R}^+$ is the set of positive real numbers plus the symbol $\top$.
Models in PEPA are built using a small but expressive set of combinators:

Prefix $(\alpha, r).C_1$. Prefix is the basic mechanism by which the behaviours of components are constructed. The component carries out activity (α, r) and subsequently behaves as component C_1.

Choice $C_1 + C_2$. The component represents a system which may behave either as component C_1 or as C_2: all the current activities of both components are enabled. A *race condition* determines the first activity to complete, and so distinguishes one component; the other is discarded.

Cooperation $C_1 \underset{L}{\bowtie} C_2$. The components proceed independently with any activities whose types do not occur in the *cooperation set* L. However, activities with action types in the set L require the simultaneous involvement of both components; these *shared activities* are only enabled when they are enabled in both C_1 and C_2. The shared activity occurs at the rate of the slowest participant. The capacity of a component C_i to carry out a given action type α (the sum of rates associated with its α actions) is called its *apparent rate*, denoted $r_\alpha(C_i)$. The apparent rate of a shared activity is the minimum, among the participating components, of the apparent rates for that type.

If an activity has rate $\top$ the component is *passive* with respect to that action type and it does not influence the rate at which such shared activities occur. When the set L is empty, we use the more concise notation $C_1 \parallel C_2$ to represent $C_1 \underset{\emptyset}{\bowtie} C_2$.

Hiding C_1/L. The component behaves as C_1 except that any activities of types within the set L are *hidden*, i.e. such an activity exhibits the unknown type τ and the activity can be regarded as an internal delay by the component. The original action type of a hidden activity is no longer accessible; the duration is unaffected.

Constant $M \stackrel{\text{def}}{=} C_1$. Constants are components whose meaning is given by a defining equation: $M \stackrel{\text{def}}{=} C_1$ gives the constant M the behaviour of the component C_1.

The semantics of PEPA, presented in the structured operational semantics style, are given in [14]. The underlying transition system also characterises the Markov process represented by the model. Rules are given for each of the combinators, showing how the component may evolve. Here we show only the rule for shared activities (Fig. 1).

$$\frac{P \xrightarrow{(\alpha, r_1)} P' \quad Q \xrightarrow{(\alpha, r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha, r)} P' \underset{L}{\bowtie} Q'} \; (\alpha \in L), \qquad r = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q))$$

Fig. 1. Operational rule defining shared activities

From a model definition M we can apply the semantic rules exhaustively to find the complete set of reachable states, the *derivative set* of M, $ds(M)$. From this set, we can construct the *derivation graph*. The derivation graph is a directed multigraph whose set of nodes is $ds(M)$ and whose arcs represent the possible transitions between them. To derive a Markov process from a PEPA model we associate a state with each node of the derivation graph. Action type information is discarded so that edges are labelled only by rates; multiple edges between a pair of nodes are combined by summing the corresponding rates. The rate on an edge in this modified graph becomes the corresponding entry in the infinitesimal generator matrix. Thus the rate between components C and C' is denoted $q(C, C')$. Similarly the conditional transition rate between C and C' due to activities of type α is denoted $q(C, C', \alpha)$.

Necessary (but not sufficient) conditions for the ergodicity of the Markov process in terms of the structure of the PEPA model have been identified and can be readily checked [14]. These imply that the model should be constructed as a cooperation of *sequential* components, i.e. components constructed using only prefix, choice and constants. Thus the compositional structure of PEPA models is at the level of the cooperating components; we refer to such models as *well-defined*. Syntactic analysis can be used to determine all the action types which will occur within the lifetime of a component C, a set denoted $\boldsymbol{\mathcal{A}}(C)$.

Example. Consider a simple two-place buffer and a server. The buffer accepts arrivals with a rate λ and passes the contents for service. When there are two customers in the buffer each attempts service, but only the front customer can be successfully serviced with rate s. Service of the second customer results in a partial service which must be corrected, at rate t before the buffer can make any other action. The server simply accepts customers for service (passively) and then allows them to depart, carrying out a false departure after a partial service. Let d be the departure rate.

In PEPA, the system is represented as the interaction of two components Buffer_0 and Server. We use three action types: *in*, *service* and *depart*. The first describes the arrival of a new customer in the buffer, the second, the service completion, and the last one the departure of a customer from the server. The components are defined as shown below.

$$\begin{array}{llll}
\mathit{Buffer}_0 \stackrel{\mathit{def}}{=} (\mathit{in}, \lambda).\mathit{Buffer}_1 & & \mathit{Server} \stackrel{\mathit{def}}{=} (\mathit{service}, \top).\mathit{Server}' \\
\mathit{Buffer}_1 \stackrel{\mathit{def}}{=} (\mathit{service}, s).\mathit{Buffer}_0 + (\mathit{in}, \lambda).\mathit{Buffer}_2 & & \mathit{Server}' \stackrel{\mathit{def}}{=} (\mathit{depart}, d).\mathit{Server} \\
\mathit{Buffer}_2 \stackrel{\mathit{def}}{=} (\mathit{service}, s).\mathit{Buffer}_1 + (\mathit{service}, s).\mathit{Buffer}_3 & & \\
\mathit{Buffer}_3 \stackrel{\mathit{def}}{=} (\mathit{service}, t).\mathit{Buffer}_1 & &
\end{array}$$

In addition to the mutually recursive sets of equations defining the behaviour of each sequential component, we have a *system equation* which defines the cooperation between the two components.

$$\mathit{System} \stackrel{\mathit{def}}{=} \mathit{Buffer}_0 \underset{\{\mathit{service}\}}{\bowtie} \mathit{Server}$$

3 Functional dependencies

In SAN, automata are able to influence one another in two ways, both related to events. Direct interaction between automata is modelled by synchronised transitions, equivalent to cooperation or shared activities in PEPA.

The other form of interaction is less direct: transition rates within an automaton can be influenced by the local states of one or more other automata of the network. Using such rates may lead to a reduction in the model size since functional rates are a means to avoid explicitly modelling all parts of a system's bahaviour. This benefit is most appreciated when building/solving the underlying Markov chain. In [15], we have introduced the notion of functional dependencies between PEPA model's components by extending the activity rates to include functional rates.

In PEPA the set of activities $\mathcal{A}ct$ is defined as $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$ where $\mathbb{R}^+$ is the set of positive real numbers defined as follows:

$$\mathbb{R}^+ = \{r | r > 0; r \in \mathbb{R}\} \cup \{\top\}$$

In the context of PEPA, a functional dependency may involve one or several components. In a functional dependency involving a single component, the rate value of one or several activities of the component depends on the current state of the component itself. This captures the presence of several apparent rates for an activity in a component. In this type of functional dependency, the rate value expressed as a function of the current component state is still a positive real number and can never be zero. However, this may not always be the case if the functional dependency involves two or more components. For example, a functional dependency between two components means that the behaviour of one component depends on the current state of the other one. This implies that either the activity to be performed by the first component and/or its rate value will be determined by the current state of the second component. The rate value may then be any non-negative real number of $\mathbb{R}^+$ including zero, particularly when the choice of the activity to be performed is done according to the state of another component.

The introduction of functional dependencies in PEPA requires us to relax the constraint on the definition domain of an activity rate [15]. Thus, the set of activities $\mathcal{A}ct$ is now defined as $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^*$ where $\mathbb{R}^*$ is the set of positive real numbers defined as follows:

$$\mathbb{R}^* = \{r | r \geq 0; r \in \mathbb{R}\} \cup \{\top\}$$

For more details about the impact of functional dependencies on PEPA models and the aggregation technique see [15].

Example. Consider again the system of the previous example. If we opt for functional rates, the buffer may then be modelled differently:

$$
\begin{aligned}
&Buffer_0 \stackrel{def}{=} (in, \lambda).Buffer_1\\
&Buffer_1 \stackrel{def}{=} (service, f \times p).Buffer_0 + (in, \lambda).Buffer_2\\
&Buffer_2 \stackrel{def}{=} (service, f \times p).Buffer_1 + (service, f \times p).Buffer_3\\
&Buffer_3 \stackrel{def}{=} (service, f \times p).Buffer_1
\end{aligned}
$$

where f is a function of the state i, $i = 0, \ldots, 3$ of component *Buffer* such that:

$$
f(i) = \begin{cases} s & \text{if} \quad i = 1 \\ t & \text{if} \quad i = 2, 3 \end{cases}
$$

and p is a probability function defined as:

$$
p(i) = \begin{cases} 1 & \text{if} \quad i = 1, 3 \\ \frac{1}{2} & \text{if} \quad i = 2 \end{cases}
$$

Note that the definition of the *Server*, and the equation defining the complete system behaviour, remain unchanged:

$$
System \stackrel{def}{=} Buffer_0 \underset{\{service\}}{\bowtie} Server
$$

This example shows that the introduction of functional rates in PEPA models allows us to avoid having different apparent rates for an activity within a single component. Whereas in the first version of the model, activity *service* has two apparent rates (s and t), the same activity has only one apparent rate (f) when using functional rates.

The association of an apparent rate with each action type within a single component leads, as we will show in Sect. 4, to a simplified tensor representation of the generator matrix associated with a PEPA model.

4 Tensor Representation

In this section we establish how to represent the infinitesimal generator matrix corresponding to a PEPA model as a sum of tensor products, analogous to the representation of a SAN. We proceed in three steps. In the first, we consider only the non-shared activities which do not belong to any cooperation set, representing the independent aspect of a component's behaviour. For each component we capture its local transitions in an appropriate matrix. In the second step, we consider the activities which belong to at least one cooperation set. This will allow us to take into account, in our tensorial representation, the interactions between the components. We represent each type of interaction by the tensor product of matrices capturing each component's capacity to participate in a shared activity. Using the obtained results, we finally show how to represent the global generator matrix of a complete model.

4.1 Non-Interacting PEPA Components

We define a non-interacting component as a component for which at least one of its activities is a non-shared activity, or for which all its activities are cooperating activities, but there exists at least one non-shared activity in the model in which this component does not participate.

With each non-interacting component C_i, $i \in \{1, .., N\}$, we associate a generator matrix R_i of size $n_i \times n_i$ with $n_i = |ds(C_i)|$. If this component has at least one non-shared activity, the elements of its matrix are the rates of its individual activities. Otherwise, the matrix associated with this component is a null matrix. In both cases, the resulting matrix describes the local transitions of the component.

Now consider a PEPA model $M \stackrel{def}{=} C_1 \parallel C_2 \parallel \ldots \parallel C_N$ and assume that C_1, C_2, ..., C_N are represented by infinitesimal generator matrices R_1, R_2, ..., R_N respectively. Then any state of M can be represented as $(C_{1,j_1}, C_{2,j_2}, \ldots, C_{N,j_N})$ where $j_i \in \{1, 2, \ldots, n_i\}$ for $1 \leq i \leq N$. Moreover, the system of the N non-interacting components may be characterised by the infinitesimal generator matrix [17]

$$
\begin{aligned}
Q = \bigoplus_{k=1}^{N} R_k &= \sum_{k=1}^{N} I_{n_1} \otimes \cdots \otimes I_{n_{k-1}} \otimes R_k \otimes I_{n_{k+1}} \otimes \cdots \otimes I_{n_N} \\
&= \sum_{k=1}^{N} \bigotimes_{i=1}^{k-1} I_{n_i} \otimes R_k \otimes \bigotimes_{i=k+1}^{N} I_{n_i}
\end{aligned}
$$

where I_d is the identity matrix of size d. $\oplus$ and $\otimes$ are the tensorial sum and product operators respectively [11].

4.2 Interacting PEPA Components

Most useful PEPA models are comprised of components which interact. To represent the interacting part of the component's behaviour, we associate with each action type α in $\mathcal{Z}$, the set of cooperating action types, a transition probability matrix $P_{i,\alpha}$. This matrix captures the capacity of component C_i to participate in the shared activity α. Thus, each element of this matrix represents the transition probability of component C_i with activity α with rate $r_\alpha(C_i)$. Note that if a component does not participate in a shared activity α, the matrix associated is an identity matrix.

In general, if a PEPA model is composed of N components, the interaction between these components can be expressed as follows:

$$
\sum_{\alpha \in \mathcal{Z}} r_\alpha \bigotimes_{i=1}^{N} P_{i,\alpha}
$$

where r_α is the minimum of the functional rates of action type α over all components C_i, $i = 1 \ldots N$:

$$r_\alpha = min(r_\alpha(C_1), r_\alpha(C_2), \ldots, r_\alpha(C_N))$$

4.3 Global Generator Matrix Representation

Now consider a PEPA model composed of interacting and non-interacting components. The corresponding generator matrix may be represented using Kronecker algebra as stated in Definition 1.

Definition 1. *The generator matrix Q of the Markov chain associated with a PEPA model is*

$$Q = \bigoplus_{i=1}^{N} R_i + \sum_{\alpha \in \mathcal{Z}} r_\alpha \left(\bigotimes_{i=1}^{N} P_{i,\alpha} - \bigotimes_{i=1}^{N} \overline{P}_{i,\alpha} \right) \tag{4.1}$$

where

- *N is the total number of components in the PEPA model and $\mathcal{Z}$ the set of cooperating action types, determined syntactically.*
- *r_α is the minimum of the functional rates of action type α over all components C_i, $i = 1 \ldots N$.*
- *R_i is the transition matrix of component C_i relating solely to its individual activities.*
- *$P_{i,\alpha}$ is the probability transition matrix of component C_i due to activity of type α. Its elements' values are between 0 and 1.*
- *$\overline{P}_{i,\alpha}$ is a matrix representing the normalization associated with the shared activity α in component C_i.*

Unlike the local transition matrices R_i, the cooperation matrices $P_{i,\alpha}$ are not generators. So we need to introduce diagonal corrector matrices $\overline{P}_{i,\alpha}$ to normalize the cooperation matrices, i.e. ensure that row sums are zero. This is shown in (4.1).

In order to apply the equation above we must place a restriction on the use of types within cooperation sets, to ensure that each action type uniquely defines a synchronisation event. To see the need for this restriction, consider the model $M \stackrel{def}{=} (V \underset{\{\alpha\}}{\bowtie} S) \parallel (T \underset{\{\alpha\}}{\bowtie} U)$. Assuming that each component enables α with just one apparent rate $r_\alpha(V)$, $r_\alpha(S)$, etc., applying the equation above, we write the generator matrix of this model as follows:

$$\begin{aligned} Q = \bigoplus_{i=1}^{4} R_i + \min(r_\alpha(V), r_\alpha(S), r_\alpha(T), r_\alpha(U)) \, (P_{V,\alpha} \otimes P_{S,\alpha} \otimes P_{T,\alpha} \otimes P_{U,\alpha} \\ - \; \overline{P}_{V,\alpha} \otimes \overline{P}_{S,\alpha} \otimes \overline{P}_{T,\alpha} \otimes \overline{P}_{U,\alpha}) \end{aligned}$$

It is clear that in this representation all the components are forced to make the *same* α cooperation. However, applying the semantics, there are two potential shared α activities: one involving V and S and one involving T and U. These can proceed concurrently with each other.

Thus we preprocess the model: when the same action type appears in distinct cooperation sets we rename the action type in the appropriate components and cooperation sets so that they are distinguished in $\mathcal{Z}$. For example, in the model above, we might distinguish α_ℓ (affecting V and S) and $\alpha_\mathfrak{r}$ (affecting T and U) and rename all α activities in V, S, T and U appropriately.

4.4 Example

Consider the two place buffer and the server described in Sect. 2. In the following we show how we construct the tensor expression for the global generator matrix of the corresponding model.

The model has two components, each component has two action types in its complete action type set: *in* and *service* for *Buffer* and *service* and *depart* for *Server*. The type *service* is the only element of $\mathcal{Z}$, the set of cooperating action types; the other action types being local to their respective components. Firstly we construct the matrices representing these local activities as follows:

$$R_{Buffer} = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -\lambda & \lambda & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R_{Server} = \begin{pmatrix} 0 & 0 \\ d & -d \end{pmatrix}$$

When we come to represent the cooperations we consider each action type in the cooperating set. In our case, this set is composed of only one action type $\alpha = service$. Component *Buffer* participates to this activity with rate $r_\alpha(Buffer) = f$ whereas component *Server* participates to this activity with rate $r_\alpha(Server) = \top$. According to the semantic of PEPA, the resulting rate r_α of the shared activity is $r_\alpha = min(f, \top) = f$. Thus the *Buffer* component's contribution and the *Server* component's contribution to the cooperation are expressed by respectively:

$$P_{Buffer,\alpha} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad \text{and} \qquad P_{Server,\alpha} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

The corresponding normalising matrix pairs are straightforward to construct:

$$\overline{P}_{Buffer,\alpha} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \overline{P}_{Server,\alpha} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Thus the complete expression becomes

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -\lambda & \lambda & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 \\ d & -d \end{pmatrix}$$

$$+ f \times \left[\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad - \quad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right]$$

leading to the complete generator matrix:

$$Q = \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 & 0 & 0 \\ d & -(d+\lambda) & 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & s & -(s+\lambda) & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & d & -(d+\lambda) & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & s & -2s & 0 & 0 & s \\ 0 & 0 & 0 & 0 & d & -d & 0 & 0 \\ 0 & 0 & 0 & t & 0 & 0 & -t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d & -d \end{pmatrix}$$

Let us consider a modification of the model in which the *Server*, instead of being passive with respect to *service* has a local rate x, such that $s < x < t$. Then the construction of the tensor expression proceeds in exactly the same way except that when we come to compute the resulting rate we obtain is $r_\alpha = min(f, x)$ which value depends on the current state of component *Buffer*. According to this, the generator matrix is:

$$Q = \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 & 0 & 0 \\ d & -(d+\lambda) & 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & s & -(s+\lambda) & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & d & -(d+\lambda) & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & \frac{x}{2} & -x & 0 & 0 & \frac{x}{2} \\ 0 & 0 & 0 & 0 & d & -d & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 & -x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d & -d \end{pmatrix}$$

4.5 Validity of the Kronecker Expression

In this section we prove the validity of the tensor expression Q given in (4.1), i.e. we show that for all reachable states the tensor expression gives us the same transition rates as the generator matrix Q^* derived from the semantics of PEPA via the labelled transition system. First, we establish some notation and terminology.

A PEPA model is given by components $(C_i)_{i\in[1..N]}$, each with state space $S_i = ds(C_i)$ and a model equation $M \stackrel{def}{=} C_1 \underset{L_1}{\bowtie} \cdots \underset{L_{N-1}}{\bowtie} C_N$. Syntactic analysis readily identifies $\mathcal{Z}$, the set of cooperating actions. For each component C_i we can identify the cooperations it must participate in, a set we denote $\mathcal{Z}_i$, where $\mathcal{Z}_i = \mathcal{Z} \cap \boldsymbol{\mathcal{A}}(C_i)$. Conversely, for each cooperating action type α, we denote by $\mathcal{Z}(\alpha)$ the set of components which participate in α typed activities.

For the same model we may have two views of the global state space. The first is $ds(M)$, the derivative set of M generated by the operational semantics via the labelled transition system. The second is $S = \prod_{i=1}^{N} S_i$, the product state space, generated directly from the derivative sets of the components. In general, the constraints placed on the model by cooperation will mean that $ds(M) \subset S$, i.e. S will contain unreachable states.

We suppose that every local state space S_i is ordered—simply take the order generated by the breadth-first search carried out in the PEPA Workbench to build the labelled transition system. In the following we assume that both $ds(M)$ and S are ordered lexicographically according to the ordering within component state spaces and the vector representation of the state space.

We will write $\mathbf{C}$ to denote a vector $(C_1, \ldots, C_N)$, and $\mathbf{C}[C_i := C_i']$ to denote the vector obtained from $\mathbf{C}$ by substituting C_i' for C_i. We denote by Q^*, the *original transition matrix* of M defined as follows:

- For all $\mathbf{C}, \mathbf{C}' \in ds(M)$ such that $\mathbf{C} \neq \mathbf{C}'$, $Q^*(\mathbf{C}, \mathbf{C}')$ is the transition rate as usually defined for PEPA, the sum of activity rates on arcs linking $\mathbf{C}$ and $\mathbf{C}'$ in the derivation graph:

$$Q^*(\mathbf{C}, \mathbf{C}') = \sum_{\mathbf{C} \xrightarrow{(\alpha,r)} \mathbf{C}'} r$$

  The set of all transitions(activities) can be partitioned into individual and shared transitions, shared transitions can be further partitioned by action type. Thus the off-diagonal elements of Q^* can be expressed as a sum of matrices as follows:

$$\begin{aligned}
Q^*(\mathbf{C}, \mathbf{C}') &= \sum_{i=1}^{N} Q_i^*(\mathbf{C}, \mathbf{C}') + \sum_{\alpha \in \mathcal{Z}} Q_\alpha^*(\mathbf{C}, \mathbf{C}') && \text{where:} \\
Q_i^*(\mathbf{C}, \mathbf{C}') &= \sum_{\mathbf{C} \xrightarrow{(\alpha,r)} \mathbf{C}'} r && \text{where } \mathbf{C}' = \mathbf{C}[C_i := {C_i}'] \text{ and } \alpha \notin \mathcal{Z} \\
Q_\alpha^*(\mathbf{C}, \mathbf{C}') &= q(\mathbf{C}, \mathbf{C}', \alpha) && \text{where } \alpha \in \mathcal{Z}
\end{aligned}$$

- For all $\mathbf{C} \in ds(M)$, $Q^*(\mathbf{C}, \mathbf{C})$ is calculated such that the sum of the elements in a row of Q^* will be zero.

Theorem 1. *Consider a well-defined PEPA model with N components interacting via a set of action types $\mathcal{Z}$ as defined by a model equation*

$$M \stackrel{\text{def}}{=} C_1 \underset{L_1}{\bowtie} \cdots \underset{L_{N-1}}{\bowtie} C_N$$

Then the original transition matrix Q^* is such that, for all reachable states, $\mathbf{C}, \mathbf{C}' \in ds(M)$, $Q^*(\mathbf{C}, \mathbf{C}') = Q(\mathbf{C}, \mathbf{C}')$ where Q is defined as

$$Q = \bigoplus_{i=1}^{N} R_i + \sum_{\alpha \in \mathcal{Z}} min(r_\alpha(C_1), \ldots, r_\alpha(C_N)) \left(\bigotimes_{i=1}^{N} P_{i,\alpha} - \bigotimes_{i=1}^{N} \overline{P}_{i,\alpha} \right)$$

Furthermore, for $\mathbf{C} \in ds(M)$ and $\mathbf{C}' \notin ds(M)$, $Q(\mathbf{C}, \mathbf{C}') = 0$, i.e. there are no transitions from reachable states to unreachable ones represented in the tensor expression.

Proof. We can re-express Q as follows:

$$Q = \sum_{i=1}^{N} G_i + \sum_{\alpha \in \mathcal{Z}} G_\alpha - \sum_{\alpha \in \mathcal{Z}} G_{\alpha,n}$$

where

$$G_i = \bigotimes_{j=1}^{i-1} I_{n_j} \otimes R_i \otimes \bigotimes_{j=i+1}^{N} I_{j_j} \tag{4.2}$$

$$G_\alpha = r_\alpha \times \bigotimes_{i=1}^{N} P_{i,\alpha} \tag{4.3}$$

$$G_{\alpha,n} = r_\alpha \times \bigotimes_{i=1}^{N} \overline{P}_{i,\alpha} \tag{4.4}$$

and $r_\alpha = min(r_\alpha(C_1), \ldots, r_\alpha(C_N))$.
First, we consider the non-diagonal elements of the matrices. We will find it convenient to use kronecker functions:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

Individual transitions. From above,

$$G_i = \bigotimes_{j=1}^{i-1} I_{n_j} \otimes R_i \otimes \bigotimes_{j=i+1}^{N} I_{n_j}$$

thus

$$G_i(\mathbf{C}, \mathbf{C}') = R_i(C_i, {C_i}') \times \prod_{\substack{k=1 \\ k \neq i}}^{N} \delta(C_k, {C_k}')$$

By the definitions of R_i and Q_i^*, it follows that, for all $\mathbf{C} \in ds(M)$, for all $i \in \{1, \ldots, N\}$,

$$G_i(\mathbf{C}, \mathbf{C}') = \begin{cases} Q_i^*(C_i, {C_i}') & \text{if } \mathbf{C}' = \mathbf{C}[C_i := {C_i}'] \\ 0 & \text{otherwise} \end{cases}$$

Clearly, $\mathbf{C}' = \mathbf{C}[C_i, {C_i}']$ implies that $\mathbf{C}'$ is reachable. Thus it follows that for $\mathbf{C}, \mathbf{C}' \in ds(M)$, for local transitions, i.e. those involving only one component C_i, the off-diagonal elements of Q_i^* and G_i are identical. Moreover, for $\mathbf{C} \in ds(M)$ and $\mathbf{C}' \notin ds(M)$, $G_i(\mathbf{C}, \mathbf{C}') = 0$.

Cooperating transitions. From above,

$$G_\alpha = r_\alpha \times \bigotimes_{i=1}^{N} P_{i,\alpha} \tag{4.5}$$

Thus

$$G_\alpha(\mathbf{C}, \mathbf{C}') = r_\alpha \times \prod_{k=1}^{N} P_{i,\alpha}(C_i, C_i')$$

Since a component i does not participate in activities of type α if $i \notin \mathcal{Z}(\alpha)$, we can rewrite this as:

$$G_\alpha(\mathbf{C}, \mathbf{C}') = r_\alpha \times \prod_{i \in \mathcal{Z}(\alpha)} P_{i,\alpha}(C_i, C_i') \times \prod_{i \notin \mathcal{Z}(\alpha)} \delta(C_i, C_i')$$

Recall that $Q_\alpha^*(\mathbf{C}, \mathbf{C}') = q(\mathbf{C}, \mathbf{C}', \alpha)$, where $\alpha \in \mathcal{Z}$. If we consider the semantic rule governing cooperation we can see that this transition rate consists of the minimal apparent rate of the participating components multiplied by the conditional probability that $\mathbf{C}'$ is the derivative resulting from the transition. This conditional probability is the product of the conditional probabilities in each component, since we assume that each component chooses between instances of α independently. For component C_i, this conditional probability is expressed as

$$p(C_i, C_i', \alpha) = \begin{cases} \frac{q(C_i, C_i', \alpha)}{q(C_i, \alpha)} & \text{if } i \in \mathcal{Z}(\alpha) \\ 1 & \text{if } i \notin \mathcal{Z}(\alpha) \text{ and } C_i = C_i' \\ 0 & \text{otherwise} \end{cases}$$

Thus, for all $\mathbf{C} \in ds(M)$

$$G_\alpha(\mathbf{C}, \mathbf{C}') = \begin{cases} r_\alpha \times p(\mathbf{C}, \mathbf{C}', \alpha) & \text{if } \mathbf{C}' \in ds(M) \\ 0 & \text{otherwise} \end{cases}$$

and so, for $\mathbf{C}, \mathbf{C}' \in ds(M)$,

$$G_\alpha(\mathbf{C}, \mathbf{C}') = r_\alpha \times p(\mathbf{C}, \mathbf{C}', \alpha)$$

It follows that for all $\mathbf{C} \in ds(M)$,

$$G_\alpha(\mathbf{C}, \mathbf{C}') = \begin{cases} Q_\alpha^*(\mathbf{C}, \mathbf{C}') & \text{if } \mathbf{C}' \in ds(M) \text{ and } \mathbf{C} \xrightarrow{(\alpha, r)} \mathbf{C}' \\ 0 & \text{otherwise} \end{cases}$$

In particular, for $\mathbf{C} \in ds(M)$ and $\mathbf{C}' \notin ds(M)$, $G_\alpha(\mathbf{C}, \mathbf{C}') = 0$ so there are no cooperating transitions into unreachable states from reachable ones.

Diagonal Elements. Finally, we consider only reachable states $\mathbf{C} \in ds(M)$ and show that $Q^*(\mathbf{C}, \mathbf{C}) = Q(\mathbf{C}, \mathbf{C})$. However, this follows immediately since by the previous arguments the off-diagonal elements of rows corresponding to $\mathbf{C}$ in Q^* and Q are in one-to-one correspondence, and furthermore, in each matrix the diagonal elements are chosen to normalise the matrix. For Q, G_i is already a generator whereas we have introduced $G_{\alpha,n}$ to normalise G_α. Thus by construction it follows that for all reachable states $\mathbf{C} \in ds(M)$, $Q^*(\mathbf{C}, \mathbf{C}) = Q(\mathbf{C}, \mathbf{C})$, as required.

□

According to the tensorial form of the generator, we store at most E entries:

$$E = (1 + 2|\mathcal{Z}|) \sum_{i=1}^{N} S_i^2$$

4.6 Solution Techniques

The tensorial representation of the generator matrix corresponding to a SAN model was proposed in 1984 by Plateau [18]. Since then different solution techniques have been investigated and several of them have been adapted to the context of this compact representation.

The main solution techniques used are either iterative methods such as the power method and Gauss Seidel or projective methods such as the Arnoldi and the GMRES methods.

In [21], the problem of computation time has been addressed. It has been shown how the methods of Arnoldi and GMRES can be used to substantially reduce the number of iterations needed when compared with the power method. Moreover, several preconditioning strategies that may be used to speed the iteration process even further have been investigated.

The power method, Arnoldi and GMRES methods have been incorporated in the tool PEPS implemented by Plateau's team. For each method, versions both with and without matrix preconditionning have been implemented.

The tensorial representation we propose for PEPA models is very similar to the one developed by Plateau for the SAN formalism. Therefore the solution techniques adapted to the tensorial representation of SAN and the computational results such as those presented in [10] about the impact of the functionnal rates on the Descriptor-vector multiplications in SAN may be, without doubt, applied in the context of PEPA formalism.

5 Related Work

Kronecker algebra representations have been used for some time as a means to address the state space explosion problem arising in the numerical solution of Markov chains. As mentioned earlier, the pioneering work in this area was carried out by Plateau on Stochastic Automata Networks [18]. More recently, Kronecker-based solution techniques have been developed for various Petri net based-formalisms, for example [8, 4–6].

With their explicit compositional structure, SPAs would appear to be natural candidates for Kronecker representation; however, there is little previous work on this topic. In 1994 Buchholz proposed an SPA called *MPA*, for which the mapping to an underlying Markov process is only defined in terms of a tensor expression [3]. However, in MPA the interpretation of both basic actions and shared actions is quite different to that in PEPA, chosen specifically to facilitate the tensor representation and without a natural modelling interpretation. MPA has not been developed further. In this approach the usual labelled transition system semantics is avoided and so there was no need to show the validity of the tensor expression with respect to the standard Markov process generation procedure. A similar denotational approach to semantics, making use of tensor expressions, is developed in the work of Rettelbach and Siegle [19].

In [9] El-Rayes presents an extension of PEPA and an associated solution technique based on the Matrix-Geometric Method (MGM). Her language PEPA$^{\infty}_{ph}$ allows exponential durations to be replaced by phase type distributions. In her mapping to the underlying Markov process, these distributions are represented by Kronecker expressions within the block-structured matrices used for the MGM. This is distinct from the use of Kronecker expressions in this paper.

6 Conclusions

In this paper we have presented a mapping from a SPA formalism to a Kronecker representation. Ours is the first such mapping aimed at implementation and incorporation into a tool. The SPA we use is PEPA and the mapping is specific to that formalism due to the complex semantic rules defining synchronisation between PEPA components. Whilst other SPAs such as EMPA [2] and IMC [12] have apparently simpler rules of synchronisation, they include immediate actions which would complicate the mapping to a Kronecker representation.

Once the prototype implementation of this approach is incorporated into the PEPA Workbench [7], we aim to improve its efficiency. In particular we plan to investigate the multilevel approaches which have been employed with SPN to avoid the incorporation of unreachable states. We will also be interested in investigating techniques which exploit this Kronecker representation to solve the model efficiently and in comparing our approach with other compact representations, such as those based on BDDs [13].

References

1. Ajmone Marsan, A., Conte, G., Balbo, G.: A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Transactions on Computer Systems **2(2)** (1984) 93–122
2. Bernardo, M., Gorrieri, R.: A Tutorial on EMPA: A theory of concurrent processes with nondeterminism, probabilities and time. Theoretical Computer Science. **201** (1998) 1–54

3. Buchholz, B.: Compositional analysis of a Markovian Process Algebra. Proc. of 2nd Process Algebra and Performance Modelling Workshop. In U. Herzog and M. Rettelbach Editors (1994)
4. Buchholz, P., Kemper, P.: Numerical analysis of stochastic marked graphs. Proc. of Int. Workshop on Petri Nets and Performance Models. IEEE-Computer Society Press. Durham, NC (1995) 32–41
5. Campos, J., Donatelli, S., Silva, M.: Structured solution of stochastic DSSP systems. Proc. of Int. Workshop on Petri Nets and Performance Models. IEEE-Computer Society Press. St Malo, France (1997) 91–100
6. Ciardo, G., Miner, A.S.: A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz editor, Proc. of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99) Zaragoza, Spain, (1999) 22–31
7. Clark, G., Gilmore, S., Hillston, J., Thomas, N.: Experiences with the PEPA performance modelling tools. IEE Software. **146(1)** (1999) 11–19
8. Donatelli, S.: Superposed Generalised stochastic Petri nets: definition and efficient solution. Proc. of 15th Int. Conf. on Application and Theory of Petri Nets. In M. Silva Editor (1994)
9. El-Rayes, A.: Analysing performance of open queueing systems with stochastic process algebra. University of Birmingham (2000)
10. Fernandes, P., Plateau, B., Stewart, W.J.: Efficient vector-descriptor multiplications in stochastic automata networks. INRIA Report #2935. Anonymous ftp. ftp.inria.fr/INRIA/Publication/RR
11. Graham, A.: Kronecker products and matrix calculus with applications. Prentice Hall (1989)
12. Hermanns, H.: Interactive Markov Chains. PhD Thesis, Universität Erlangen-Nürnberg (1999)
13. Hermanns, H., Meyer-Kayser, J., Siegle, M.: Multi-terminal binary decision diagrams to represent and analyse continuous time Markov chains. Proc. 3rd Int. Workshop on Numerical Solution of Markov Chain, Zaragoza, Spain, (1999) 188–207
14. Hillston, J.: A Compositional approach to performance modelling. PhD Thesis, The University of Edinburgh (1994)
15. Hillston, J., Kloul, L.: From SAN to PEPA: A technology transfer. Submitted.
16. Molloy, M.K.: Performance analysis using stochastic Petri nets. IEEE Transactions on Computers **31(9)** (1982) 913–917
17. Plateau, B.: On the stochastic structure of parallelism and synchronisation models for distributed algorithms. Proc. ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems (1985)
18. Plateau, B.: De l'évolution du parallélisme et de la synchronisation. PhD Thesis, Université de Paris-Sud, Orsay (1984)
19. Rettelbach, M., Siegle, M.: Compositional minimal semantics for the stochastic process algebra TIPP. Proc. of 2nd Workshop on Process Algebra and Performance Modelling (1994)
20. Sanders, W.H., Meyer, J.F.: Reduced base model construction methods for stochastic activity networks. IEEE Journal on Selected Areas in Communications **9(1)** (1991) 25–36
21. Stewart, W.J., Atif, K., Plateau, B.: The numerical solution of stochastic automata networks. European Journal of Operations Research **86(3)** (1995) 503–525