Compositional Abstractions for Long-Run Properties of Stochastic Systems

Michael J. A. Smith
Department of Informatics and Mathematical Modelling
Danmarks Tekniske Universitet
Lyngby, Denmark
Email: mjas@imm.dtu.dk

***Abstract*—When analysing the performance of a system, we are often interested in long-run properties, such as the proportion of time it spends in a certain state. Stochastic process algebras help us to answer this sort of question by building a *compositional* model of the system, and using tools to analyse its underlying Markov chain. However, this also leads to state space explosion problems as the number of components in the model increases, which severely limits the size of models we can analyse. Because of this, we look for *abstraction* techniques that allow us to analyse a smaller model that safely bounds the properties of the original.**

In this paper, we present an approach to bounding long-run properties of models in the stochastic process algebra PEPA. We use a method called *stochastic bounds* to build upper and lower bounds of the underlying Markov chain that are lumpable, and therefore can be reduced in size. Importantly, we do this *compositionally*, so that we bound each component of the model separately, and compose these to obtain a bound for the entire model. We present an algorithm for this, based on extending the algorithm by Fourneau *et al*. to deal with partially-ordered state spaces. Finally, we present some results from our implementation, which forms part of the PEPA plug-in for Eclipse. We compare the precision and state space reduction with results obtained by computing long-run averages on a CTMDP-based abstraction.

I. Introduction

The primary aim of stochastic modelling is to gain insight and understanding of systems that arise in practice, but the models we create of such 'real' systems are often much too large to analyse explicitly. We therefore need techniques for *abstracting* these models — specifically, reducing their size so that they become small enough to analyse. Given that we are interested in some particular properties of the model, we need to ensure that the abstract model gives us accurate results, but this usually comes at the expense of lower precision. For example, if the property is a probability, the abstract model could give the interval $[0.1, 0.2]$, which is accurate, but less precise than that actual answer of 0.18.

In this paper, we will consider long-run properties of Markovian models — after the system has been running for a long time, what proportion of time does it spend in a particular set of states? By answering this, we can calculate important performance characteristics such as throughput, utilisation and identification of bottleneck components.

To allow us to analyse such properties for large models, we present a *compositional* approach to abstraction, based on the technique of *stochastic bounds* [18]. The idea is to construct an upper (or lower) bound of a Markov chain that is *lumpable* with respect to a given abstraction — i.e. we can aggregate states to build a smaller Markov chain that preserves the long-run properties of the original. An algorithm for doing this was presented in [9], and we extend this so that it can be applied compositionally and over partially-ordered state spaces.

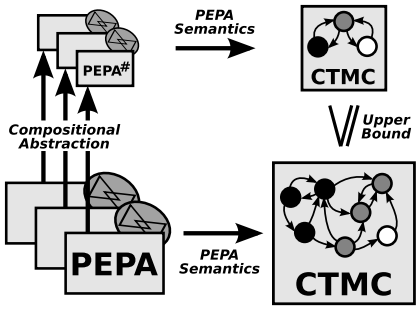

Fig. 1. Compositional abstraction of PEPA models

Our framework for compositional abstraction is presented for the stochastic process algebra PEPA [11] — although our results could be applied to other stochastic process algebras with CSP-style synchronisation. We extend the notion of a stochastic bound so that it can be applied to models compositionally. This is illustrated in Figure 1 — we bound and aggregate each component in a PEPA model separately, such that the aggregated model induces a Markov chain that is an upper bound of the original. This means that we *do not need to construct* the state space of the original Markov chain; only the abstracted one. To do this, we use a Kronecker representation for the underlying Markov chain of a PEPA model, which was presented in [17]. Importantly, we have also implemented the algorithm in this paper as part of the model checker in the PEPA plug-in for Eclipse [1], [16].

There has been other work in relation to compositional applications of stochastic bounds [5], [19]. In particular, [5] uses a weaker constraint than stochastic monotonicity, to obtain tighter bounds. Our approach differs by working with stochastic monotonicity over partially ordered state spaces.

A summary of this paper is as follows. We begin in Section II by introducing the use of Markov chains and PEPA for stochastic modelling. We then introduce abstraction techniques for Markov chains, and in particular stochastic bounds, in Section III. In Section IV we show how to compositionally

apply stochastic bounds to PEPA models, and we present an algorithm for computing bounds from a partially ordered state space in Section V. Finally, we demonstrate this technique on an example model in Section VI, before concluding in Section VII. This is an extended version of the paper published in QEST 2011 (copyright IEEE), with proofs contained in the attached appendix.

II. Stochastic Modelling

When we build stochastic models of systems, we often work with *Markov chains*, since there exist efficient numerical solution techniques. These come in two flavours — *discrete time* Markov chains, where we model how the system evolves, but not how long it takes to do so, and *continuous time* Markov chains, which are enriched with a notion of time.

Definition 1. *A* Discrete Time Markov Chain (DTMC) *is a tuple $(S, \boldsymbol{P}, L)$, where S is a finite non-empty set of states, $\boldsymbol{P} : S \times S \to [0,1]$ assigns a probability distribution over S to each state $s \in S$, and $L : S \to AP$ is a labelling function (AP is a finite set of atomic propositions). We require for all $s \in S$ that $\sum_{s' \in S} \boldsymbol{P}(s,s') = 1$.*

Definition 2. *A* Continuous Time Markov Chain (CTMC) *is a tuple $(S, r, \boldsymbol{P}, L)$, where $(S, \boldsymbol{P}, L)$ is a DTMC (the embedded DTMC), and $r : S \to \mathbb{R}_{\geq 0}$ is a function describing the rate of exit for each state, such that $r(s) = 0$ iff $\boldsymbol{P}(s,s) = 1$. We write $Embed(M)$ for the embedded DTMC of a CTMC M.*

In a CTMC, the time spent in a state s before making a transition (the sojourn time) is governed by an exponentially-distributed random variable X_s, such that for all $t \in \mathbb{R}_{\geq 0}$, $\mathbf{Pr}(X_s \leq t) = 1 - e^{-r(s)t}$.

In the above definitions, $\boldsymbol{P}$, r and L are functions, but it is often convenient to think of them instead as *matrices* ($\boldsymbol{P}$) and *vectors* (r and L). This requires us to have an *ordering* on the state space S, so that we map a state onto an index. For the presentation in this section, the particular ordering is not important, but it will play a major role when we look at bounding techniques in Section III.

It is often useful for us to obtain a DTMC from a CTMC, since certain properties are easier to check. If we just take the embedded DTMC, however, we lose the fact that may we spend longer on average in some states than in others — i.e. that some states have different exit rates. A useful transformation is to *uniformise* the CTMC, so that all the exit rates are the same. We do this by adding self-loops to states:

Definition 3. *The* uniformisation *of a CTMC $\mathcal{M} = (S, r, \boldsymbol{P}, L)$ with rate $\lambda \geq \max_{s \in S} r(s)$ is $Unif_\lambda(\mathcal{M}) = (S, \overline{r}, \overline{\boldsymbol{P}}, L)$, where $\overline{r}(s) = \lambda$ for all $s \in S$, and:*

$$\begin{aligned} \overline{\boldsymbol{P}}(s,s') &= \tfrac{r(s)}{\lambda}\boldsymbol{P}(s,s') && \text{if } s \neq s' \\ \overline{\boldsymbol{P}}(s,s) &= 1 - \textstyle\sum_{s' \neq s} \overline{\boldsymbol{P}}(s,s') && \text{otherwise} \end{aligned}$$

This preserves all properties of the original CTMC that are not sensitive to the distinction between making a transition to the same state, and just staying in a state without exiting it.

To be more precise about what we mean by a *property*, there are two types of quantitative properties that people usually consider. *Transient properties* concern the behaviour of the Markov chain up to the time at which some event happens (e.g. entering a certain state). *Long-run properties*, on the other hand, concern the behaviour of the model in the limit as it runs forever — in particular, the *long-run average* proportion of time that we spend in a certain set of states.

To define such long-run averages for a Markov chain, it is convenient to use the labelling function to identify the states that we are interested in. Letting $AP = \{0, 1\}$, we can identify a set of states by the label $L(s) = 1$. The long-run average $\rho(s)$ of being in states s' for which $L(s') = 1$ (starting from state s) is given by the limit:

$$\rho(s) = \lim_{n \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} \sum_{s' \in S} \boldsymbol{P}^i(s,s') L(s')$$

If a Markov chain is *ergodic* — that is to say, it has a strongly-connected state space (for a CTMC), or all its states are positive recurrent and aperiodic (for a DTMC) — then we can also express the long-run average $\rho(s)$ in terms of the *steady state distribution* $\boldsymbol{\pi}$: for a DTMC, this is the unique solution to the equation system $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ subject to the constraint $\sum_{s \in S} \boldsymbol{\pi}(s) = 1$. We can then compute $\rho(s) = \sum_{s' \in S} \boldsymbol{\pi}(s') L(s')$.

Note that even if a CTMC is not ergodic, we can still compute long-run average probabilities using this approach — we find the bottom strongly-connected components (BSCCs) in the CTMC, and compute the steady state distribution for each of these, along with the probability of reaching each BSCC from a given initial state [3].

A. The Performance Evaluation Process Algebra (PEPA)

When we build a performance model of a system, we usually want to use a *modelling language*, rather than writing down a Markov chain directly. To this end, stochastic process algebras have been developed as *compositional* modelling formalisms — we model a number of components individually, which are then composed together. The Performance Evaluation Process Algebra (PEPA) [11] is one such language, whose semantics maps a model onto a CTMC.

The syntax of PEPA is as follows:

$$\begin{aligned} C_S &:= (a, r).C_S \mid C_S + C_S \mid A \\ C_M &:= C_S \mid C_M \underset{\mathcal{L}}{\bowtie} C_M \mid C_M/\mathcal{L} \end{aligned}$$

A *sequential component* C_S describes an individual component in the model, and a *model component* C_M describes an entire PEPA model. Components can perform and synchronise over *activities*, which have an associated *action* (from a finite set Act), along with a *rate* that describes its duration. The meaning of each of the PEPA combinators is as follows:

- *Prefix* ($(a, r).C$): the component carries out an activity with action a at rate $r \in \mathbb{R}_{\geq 0} \cup \{\top\}$, to become C. If $r = \top$, the activity is *passive*, and we need to synchronise with another component to determine the rate.

- *Choice* ($C_1 + C_2$): the component makes a choice between behaving as C_1 or as C_2. This choice is made by a race condition on the enabled activities of these components — the first activity to complete determines which component proceeds, with the other being discarded.
- *Cooperation* ($C_1 \underset{\mathcal{L}}{\bowtie} C_2$): the components C_1 and C_2 synchronise over the actions in $\mathcal{L}$. An activity (a, r_1) of C_1 with $a \in \mathcal{L}$ can only proceed if C_2 also performs an activity (a, r_2) — they proceed together, at the rate of the slowest component. Otherwise, if $a \notin \mathcal{L}$, the activities take place independently.
- *Hiding* ($C/\mathcal{L}$): the component behaves as C, except that any actions in $\mathcal{L}$ are renamed to the hidden type τ, which cannot be synchronised over.
- *Constant* ($A \stackrel{def}{=} C$): the component C is named A.

If we ignore the hiding operator[1], a PEPA model has the following structure, where C_i are sequential components:

$$C_1 \underset{\mathcal{L}_1}{\bowtie} \cdots \underset{\mathcal{L}_{N-1}}{\bowtie} C_N \tag{1}$$

An operational semantics of PEPA is given in [11], which maps a model onto a labelled multi-transition system, from which we can derive a CTMC. This approach considers the model as a whole, however, and so is not compositional. Instead, we will define the PEPA semantics in a component-wise fashion, using a *Kronecker representation* for the CTMC of a PEPA model. This was first introduced in [12], but we will use a variant that was developed in [17].

To build the CTMC of a PEPA model compositionally, it is useful to introduce the notion of a *CTMC component*:

Definition 4. *A* CTMC component *is a tuple* $(S, r, \boldsymbol{P}, L)$*, where* S*,* $\boldsymbol{P}$ *and* L *are defined as for a CTMC, and* $r : S \to \mathbb{R}_{\geq 0} \cup \{ \top \}$ *assigns a rate (or* $\top$*) to each state.*

Note that if none of the rates are passive, a CTMC component corresponds to a CTMC.

We will use two composition operators for CTMC components — $\circledast$ and $\odot$ — corresponding to synchronised and independent parallel composition respectively. For $M_1 = (S_1, r_1, \boldsymbol{P}_1, L_1)$ and $M_2 = (S_2, r_2, \boldsymbol{P}_2, L_2)$:

$$M_1 \circledast M_2 = (S_1 \times S_2, \min\{ r_1, r_2 \}, \boldsymbol{P}_1 \otimes \boldsymbol{P}_2, L_1 \times L_2)$$

$$\begin{aligned} M_1 \odot M_2 = {} & (S_1, r_1, \boldsymbol{P}_1, L_1) \circledast (S_2, r'_\top, \boldsymbol{I}, L_2) \\ & + (S_1, r'_\top, \boldsymbol{I}, L_1) \circledast (S_2, r_2, \boldsymbol{P}_2, L_2) \end{aligned}$$

where $\min\{ r_1, r_2 \}(s_1, s_2) = \min\{ r_1(s_1), r_2(s_2) \}$, $r_\top(s) = \top$ for all s, and $\boldsymbol{I}(s_1, s_2) = 1$ if $s_1 = s_2$ and 0 otherwise. The operator $\otimes$ denotes the Kronecker product of two matrices [15]. Recalling that $AP = \{ 0, 1 \}$, we define $(L_1 \times L_2)(s_1, s_2) = L_1(s_1) \cdot L_2(s_2)$.

For two CTMC components $M_1 = (S, r_1, \boldsymbol{P}_1, L)$ and $M_2 = (S, r_2, \boldsymbol{P}_2, L)$ with the same state space S and labelling function L, we define their addition as follows:

$$M_1 + M_2 = \left(S, r_1 + r_2, \frac{r_1}{r_1 + r_2} \boldsymbol{P}_1 + \frac{r_2}{r_1 + r_2} \boldsymbol{P}_2, L \right)$$

[1]We can do this without loss of generality, since it is always possible to rename action types to avoid name conflicts between components.

for $(r_1 + r_2)(s) = r_1(s) + r_2(s)$, and $\frac{r_i}{r_1 + r_2}(s) = \frac{r_i(s)}{r_1(s) + r_2(s)}$, $i \in \{ 1, 2 \}$, for all $s \in S$, and $(r\boldsymbol{P})(s_1, s_2) = r(s_1)\boldsymbol{P}(s_1, s_2)$.

We can now describe the semantics of PEPA. Using the operational semantics in [11], we can derive a CTMC component $[\![C]\!]^{PEPA} = (S, r, \boldsymbol{P}, L)$ from a PEPA sequential component C (we assume that we have some way of specifying the labelling function L for the component). We write $[\![C]\!]^{PEPA}_a = (S, r_a, \boldsymbol{P}_a, L)$ for the CTMC component over the same state space S, but where r_a and $\boldsymbol{P}_a$ contain only the transitions corresponding to activities of action type a.

Definition 5. *The CTMC induced by a PEPA model* C *is:*

$$[\![C]\!] = \sum_{a \in Act(C)} [\![C]\!]_a$$

where $Act(C)$ *is the set of all action types that occur in* C *(both synchronised and independent), and* $[\![C]\!]_a$ *is as follows (S.C. stands for Sequential Component):*

$$\begin{aligned} [\![C]\!]_a &= [\![C]\!]^{PEPA}_a && \text{if } C \text{ is an S.C.} \\ [\![C_1 \underset{\mathcal{L}}{\bowtie} C_2]\!]_a &= \begin{cases} [\![C_1]\!]_a \circledast [\![C_2]\!]_a & \text{if } a \in \mathcal{L} \\ [\![C_1]\!]_a \odot [\![C_2]\!]_a & \text{if } a \notin \mathcal{L} \end{cases} \end{aligned}$$

The proof that this construction corresponds to the CTMC induced by the PEPA semantics is given in [17]. More precisely, the CTMCs given by the reachable state spaces of both semantics, from the same initial state, are isomorphic.

III. Abstractions of Markov Chains

When we build a performance model of any reasonably large system, we often run into difficulty with the size of the model. This is particularly true when we use a compositional formalism such as PEPA, where we run into the state space explosion problem. One approach to analysing such large models is to *abstract* them — that is to say, we construct a smaller model that is easier to analyse, but that preserves some properties of the original model.

To abstract a Markov chain, we need to map its state space S onto a smaller state space $S^\sharp$. This can be done by means of a surjective function $\alpha : S \to S^\sharp$ that maps every state in S onto one in $S^\sharp$. We call $(S^\sharp, \alpha)$ an *abstraction* of S. If we try to apply this to a CTMC, we have a problem defining the new rates and transition probabilities, since we might map states with different behaviour onto the same abstract state. In fact, the abstraction typically only yields a CTMC if it satisfies a condition called *ordinary lumpability* [14] (also known as *strong probabilistic bisimulation*)[2]:

Definition 6. *An* ordinarily lumpable *abstraction* $(S^\sharp, \alpha)$ *of a CTMC* $\mathcal{M} = (S, r, \boldsymbol{P}, L)$ *is one such that for all states* $s, s' \in S$*, if* $\alpha(s) = \alpha(s')$ *then* $L(s) = L(s')$ *and for all states* $s^\sharp \in S^\sharp$*:*

$$\sum_{\{ t \mid \alpha(t) = s^\sharp \}} r(s)\boldsymbol{P}(s, t) = \sum_{\{ t \mid \alpha(t) = s^\sharp \}} r(s')\boldsymbol{P}(s', t)$$

[2]It has also been shown that Markovian testing equivalence and Markovian trace equivalence induce exact aggregations [4].

The same definition holds for DTMCs if we remove the occurrences of the rate function r. An ordinarily lumpable abstraction *induces* a lumped CTMC by defining the new transition probabilities and exit rates.

Unfortunately, it is usually *not* the case that an abstraction of a Markov chain is ordinarily lumpable. One approach, such as in [6], is to move to Markov Decision Processes (MDPs), which have both probabilities (and rates in the case of continuous-time MDPs) and *non-determinism*. A similar approach is taken in [13], which uses *abstract Markov chains* (also known as interval Markov chains), where there are *intervals* of probabilities in the abstraction.

The problem with these methods is that by introducing non-determinism, we lose the ability to compute steady state probabilities. One approach is to use techniques for directly computing long-run averages on the MDP abstraction, such as in [7] and more recently in [20]. In this paper, we take an alternative view, using *stochastic bounds* [18] to construct an abstraction that is *still a Markov chain*, allowing the standard approach to calculating long-run averages (based on computing the steady state distribution of BSCCs) to be applied [3]. We will return briefly to the former approach as a comparison, in Section VI.

Stochastic bounding allows us to construct upper and lower bounds for monotone properties of Markov chains. Since the focus of this paper is on long-run average properties, we will look at bounds on the steady-state distribution of ergodic Markov chains. The important idea is to construct bounding Markov chains that are *lumpable*, so that we can reduce the size of the Markov chain to solve. But to formally define what we mean by a *bound*, we first need to introduce a *stochastic ordering* for probability distributions.

There are a number of different stochastic orderings [18], but for our purposes we will use only the *strong stochastic order*, which we denote $\leq_{\mathrm{st}}$. This is defined for comparing random variables in general, but for our purposes we will consider just discrete random variables, which can be represented as a vector of probabilities, summing to one:

Definition 7. *Let X be a random variable on a partially ordered state space $(S, \prec)$, and $\boldsymbol{x}$ be a vector such that $\mathbf{Pr}(X \succ s) = \sum_{s' \succ s} \boldsymbol{x}(s')$. Let $\boldsymbol{y}$ be defined similarly for a random variable Y over $(S, \prec)$. We say that $\boldsymbol{x} \leq_{\mathrm{st}} \boldsymbol{y}$ if for all $s \in S$:*

$$\sum_{s' \succ s} \boldsymbol{x}(s') \leq \sum_{s' \succ s} \boldsymbol{y}(s')$$

We can extend this ordering to Markov chains as follows. The DTMCs $M_1 = (S, \boldsymbol{P}_1, L)$ and $M_2 = (S, \boldsymbol{P}_2, L)$ are comparable ($M_1 \leq_{\mathrm{st}} M_2$) with respect to an initial distribution $\boldsymbol{\iota} : S \to [0, 1]$ if for all $n \in \mathbb{N}$, $\boldsymbol{\iota}\boldsymbol{P}_1^n \leq_{\mathrm{st}} \boldsymbol{\iota}\boldsymbol{P}_2^n$. This is not a very practical definition, however, and so we instead look for conditions based on properties of $\boldsymbol{P}$.

There are two important properties of such matrices: *comparability* and *monotonicity*. We shall assume that the state space of a matrix $\boldsymbol{P}$ (i.e. its row and column indices) is a partially ordered set $(S_P, \prec_P)$, and we shall omit the subscript when it is clear from context. Furthermore, we use the notation $\boldsymbol{P}(i, *)$ for row i of matrix $\boldsymbol{P}$, which is itself a row vector.

Definition 8. *A stochastic matrix $\boldsymbol{P}$ is* monotone *if for all row vectors $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{u} \leq_{\mathrm{st}} \boldsymbol{v}$ implies that $\boldsymbol{u}\boldsymbol{P} \leq_{\mathrm{st}} \boldsymbol{v}\boldsymbol{P}$. Equivalently, for all $s, s' \in S$ such that $s \prec s'$, $\boldsymbol{P}(s, *) \leq_{\mathrm{st}} \boldsymbol{P}(s', *)$.*

Definition 9. *The stochastic matrices $\boldsymbol{P}$ and $\boldsymbol{P}'$ are* comparable*, denoted $\boldsymbol{P} \leq_{\mathrm{st}} \boldsymbol{P}'$, if they share the same state space $(S, \prec)$, and for all $s \in S$, $\boldsymbol{P}(s, *) \leq_{\mathrm{st}} \boldsymbol{P}'(s, *)$.*

Theorem 10. *Consider the DTMCs $\mathcal{M}_1 = (S, \boldsymbol{P}_1, L)$ and $\mathcal{M}_2 = (S, \boldsymbol{P}_2, L)$ with the same state space S and labelling function L. The following requirements are sufficient to ensure that $\mathcal{M}_1 \leq_{\mathrm{st}} \mathcal{M}_2$ for all initial distributions:*

1) *$\boldsymbol{P}_1 \leq_{\mathrm{st}} \boldsymbol{P}_2$.*
2) *At least one of $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ is monotone.*

Theorem 11. *Two CTMCs $\mathcal{M}_1 = (S, \boldsymbol{P}_1, r_1, L)$ and $\mathcal{M}_2 = (S, \boldsymbol{P}_2, r_2, L)$ are comparable such that $\mathcal{M}_1 \leq_{\mathrm{st}} \mathcal{M}_2$ if for all $\lambda \geq \max\{\, \max_{s \in S} r_1(s), \max_{s \in S} r_2(s) \,\}$:*

$$\mathit{Embed}(\mathit{Unif}_\lambda(\mathcal{M}_1)) \leq_{\mathrm{st}} \mathit{Embed}(\mathit{Unif}_\lambda(\mathcal{M}_2))$$

Since stochastic comparison of CTMCs is defined in terms of stochastic comparison of DTMCs, we can consider only the latter, without loss of generality, for the remainder of this section. The need to uniformise CTMCs, however, will become important when we apply these techniques compositionally to PEPA models in Section IV.

For stochastic bounds to be of practical use, we need algorithms to construct monotone upper and lower bounding matrices, given the probability transition matrix of a DTMC. Furthermore, not only do we need them to be bounding, but they must be *lumpable* with respect to the desired abstraction.

In [9], an algorithm is given to derive an irreducible and lumpable bounding matrix for a DTMC, assuming a *totally ordered* state space. This is an extension of the algorithm in [2], which just finds a monotone upper-bounding probability transition matrix for a DTMC. This algorithm arose from observing that, for $\boldsymbol{P} \leq_{\mathrm{st}} \boldsymbol{R}$, the following inequalities must hold for the $(n \times n)$ stochastic matrix $\boldsymbol{R}$ to be monotone:

1) For all $1 \leq i \leq n$, $\boldsymbol{P}(i, *) \leq_{\mathrm{st}} \boldsymbol{R}(i, *)$.
2) For all $1 \leq i \leq n-1$, $\boldsymbol{R}(i, *) \leq_{\mathrm{st}} \boldsymbol{R}(i+1, *)$.

In the basic algorithm, the first row of $\boldsymbol{R}$ is equal to that of $\boldsymbol{P}$, and each subsequent row is determined by the maximum of the left-hand sides of the above inequalities. The first condition can be restated as $\sum_{k=j}^{n} \boldsymbol{P}(i, k) \leq \sum_{k=j}^{n} \boldsymbol{R}(i, k)$, for all j, since the ordering is total. Hence:

$$\boldsymbol{R}(i, j) = \max \left\{ \sum_{k=j}^{n} \boldsymbol{R}(i-1, k), \sum_{k=j}^{n} \boldsymbol{P}(i, k) \right\} - \sum_{k=j+1}^{n} \boldsymbol{R}(i, k)$$

We can iteratively construct the matrix $\boldsymbol{R}$ based on the above equation, starting with the first row and the last column, and iterating down the rows before moving onto the next column. Since the algorithm for computing a lower bounding

matrix is very similar, we will only consider the construction of upper bounds throughout this paper.

Unfortunately, this does not guarantee that if $\boldsymbol{P}$ is irreducible then $\boldsymbol{R}$ will also be, since it is possible to delete transitions. Fourneau *et al*. [9] address this by modifying the algorithm to avoid unnecessarily deleting transitions. Furthermore, they produce an upper-bounding matrix that is not only monotone and irreducible, but *lumpable* with respect to a given partition. To achieve this, they structure the matrix so that states in the same partition have contiguous indices, and make the matrix lumpable by setting the next state distribution in each partition to the maximum that occurs within that partition. This ensures that the matrix remains monotone. The worst-case time complexity of this algorithm is $O(n^2)$, where n is the size of the state space (i.e. $\boldsymbol{P}$ and $\boldsymbol{R}$ are $n \times n$ matrices) [9].

The main disadvantage of this algorithm is that it requires the state space to be totally ordered. It is possible to extend any partial order to a total order, but this will result in a stronger order than we need, and so give lower precision in the bounds. We will see later, in Section V, how to modify this algorithm to work with a simple class of partially ordered state spaces.

IV. Stochastic Bounding of PEPA Models

Having introduced the Kronecker form for PEPA models, we can turn to applying stochastic bounds compositionally. In this section, we will extend the definitions of stochastic ordering and monotonicity, so that when we compose the bounds of two PEPA components, the resulting CTMC is a bound of that induced by the original components. We present an algorithm for constructing these bounds in Section V.

The work we present here is general in the sense that it applies to *all* PEPA models. This is in contrast to previous work, which considers the application of stochastic bounds to particular classes of PEPA model, such as passage time properties of workflow-structured models [8]. There is an advantage to looking at specific classes of models, in that it may be possible to obtain more precise bounds in light of the additional information that is available. However, generality is also important, in that we can analyse models without needing to assume anything about their particular structure.

Before bounding a PEPA model, we need to decide upon two things — an *ordering*, and a *partitioning* of its state space. Since the idea is to produce the bound compositionally, these must also be defined compositionally. If we have an abstraction $(S_i^\sharp, \alpha_i)$ for each component C_i in a PEPA model, we can compose these to induce an abstraction for the entire model. This defines a unique partitioning of the state space according to which concrete states map to the same abstract state.

In addition to partitioning the state space, we need to provide an ordering. The ordering we choose will in general depend on the property we are interested in. For example, if are interested in the steady state probability of being in a particular set of states, it makes sense to place these at the 'top' of the ordering. This is so that we can directly compare the probability of being in this set. Furthermore, choosing a partial order can be advantageous, since it allows more flexibility

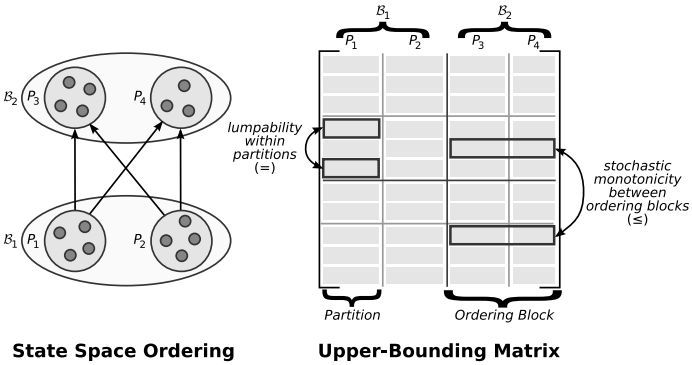

Fig. 2. State space ordering and lumpability constraints

when constructing the bound. The only constraint we have is to only allow entire partitions to be compared with other partitions, so that the abstraction can be applied.

The definitions and theorems in this section are applicable to any partial order, but in order to algorithmically construct the bound, we will restrict ourselves to the following class:

Definition 12. *A simple partial order over a state space S is given by a set of M disjoint sets, $\mathcal{B}_1, \ldots, \mathcal{B}_M \subseteq S$, where:*

$$s \prec s' \quad \textit{iff} \quad \exists i, j.\ s \in \mathcal{B}_i \wedge s' \in \mathcal{B}_j \wedge i < j$$

Each $\mathcal{B}_i$ may contain multiple partitions, but not vice versa. This is illustrated in Figure 2, which shows an example state space ordering and partitioning, and the resulting constraints on the upper-bounding transition matrix.

Given an ordering and partitioning of a state space, we need to find a monotone CTMC that is both lumpable and an upper bound of the original CTMC. To do this compositionally, we must work at the level of CTMC components, bounding both the rate function r and the transition matrix $\boldsymbol{P}$ separately. This is so that when we construct the CTMC of the entire model, according to Definition 5, it remains upper-bounding, monotone and lumpable.

Unfortunately, it is not necessarily the case that a CTMC is monotone, even if r and $\boldsymbol{P}$ both are. Recall that for a CTMC to be monotone we require its embedded DTMC after uniformisation to be monotone. This is illustrated by the following example — r and $\boldsymbol{P}$ are monotone, but $\overline{\boldsymbol{P}}$ is not:

$$M = (S, r, \boldsymbol{P}, L) = \left\langle S, \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{5} & 0 & \frac{1}{5} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}, L \right\rangle$$

$$\mathit{Unif}_2(M) = (S, \overline{r}, \overline{\boldsymbol{P}}, L) = \left\langle S, \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}, L \right\rangle$$

To avoid this problem, we need to strengthen the definitions of stochastic ordering and monotonicity, by adding an extra constraint. We call these the *rate-wise stochastic ordering* and *rate-wise monotonicity* respectively. Their definitions are:

Definition 13. *Two CTMC components $M = (S, r, \boldsymbol{P}, L)$ and $M' = (S, r', \boldsymbol{P}', L)$, with the same state space S and*

labelling function L, are ordered $M \leq_{\mathrm{rst}} M'$ under the rate-wise stochastic ordering*, if:*

1) $\boldsymbol{P} \leq_{\mathrm{st}} \boldsymbol{P}'$
2) *For all states $s \in S$ such that $r(s) \neq r'(s)$, if $r(s) = 0$ and $r'(s) > 0$ then $\forall s' \prec s. \sum_{t \succ s'} \boldsymbol{P}'(s,t) = 1$, otherwise:*

$$1 \leq \frac{r'(s)}{r(s)} \leq \min_{s' \prec s} \left\{ \frac{1 - \sum_{t \succ s'} \boldsymbol{P}(s,t)}{1 - \sum_{t \succ s'} \boldsymbol{P}'(s,t)} \right\}$$

Definition 14. *A CTMC component $M = (S, r, \boldsymbol{P}, L)$ is rate-wise monotone if:*

1) $\boldsymbol{P}$ *is monotone.*
2) *For all states $s, s' \in S$ such that $s \prec s'$ and $r(s) \neq r(s')$, if $r(s) = 0$ and $r(s') > 0$ then $\forall s'' \prec s. \sum_{t \succ s''} \boldsymbol{P}(s',t) = 1$, otherwise:*

$$1 \leq \frac{r(s')}{r(s)} \leq \min_{s'' \prec s} \left\{ \frac{1 - \sum_{t \succ s''} \boldsymbol{P}(s,t)}{1 - \sum_{t \succ s''} \boldsymbol{P}(s',t)} \right\}$$

Intuitively, condition 2 ensures that the probability transition matrix increases faster than the rate function, so that after uniformisation we remain monotone and comparable. Note that this is not a condition on the original model — we just need to ensure that it holds for the upper bound.

We can show that strong stochastic comparison and monotonicity follow from rate-wise stochastic comparison and monotonicity. This means that the CTMC that we construct by this method is stochastically comparable in the usual sense. We state this in the following two theorems:

Theorem 15. *If $M = (S, r, \boldsymbol{P}, L) \leq_{\mathrm{rst}} M' = (S, r', \boldsymbol{P}', L)$ and for all $s \in S$, $r(s) \leq r'(s)$, then $M \leq_{\mathrm{st}} M'$.*

Theorem 16. *If $M = (S, r, \boldsymbol{P}, L)$ is rate-wise monotone, and for all $s \prec s' \in S$, $r(s) \leq r(s')$, then M is monotone.*

Unfortunately, it is still not the case that rate-wise monotonicity and rate-wise stochastic ordering are preserved in general when two components cooperate. The problem arises due to the minimum operator, which is applied to the rate functions. If we take monotonicity, for example, the ratio between successive rates places constraints on the probabilistic transition matrix. When we compose two monotone components, it is possible for one to be completely bounded by the other in terms of its ability to perform an activity of type a. That is to say, the rate of performing a in each state of one component is less than the rate of a in *any* state of the other. Hence the minimum of the two rate functions, and the resulting constraint on the composed probabilistic transition matrix, depends on only one of the components. The required constraint on the composed matrix may therefore be tighter than that of one of the components.

This problem is clearer if we look at a particular example:

$$M = (S, r, \boldsymbol{P}, L) = \left(S, \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}, L \right)$$

This CTMC component *is* rate-wise monotone, but if it were to synchronise with a component that has a rate function $r = [1, 1, 2]^{\mathsf{T}}$, we would result in the same problem as before.

It is therefore not possible for us to construct a bound for a sequential component, without considering the *context* in which it occurs. To define this context, we need a measure on components, to indicate the extent to which the rate function increases. For monotonicity, we are concerned with the ratio between successive rates, and in particular the maximum of these. This is because, when taking the Kronecker product, we consider all possible state combinations. Hence the maximum increase will actually occur, and gives a bound on how a component can affect those that it cooperates with.

Definition 17. *The internal rate measure of a PEPA component C, where $[\![C]\!]_a = (S, r_a, \boldsymbol{P}_a, L)$ for action type a, is:*

$$\|C\|_a = \begin{cases} \top \text{ if } \exists s.\ \mathrm{succ}(s) \neq \emptyset \wedge r_a(s) = 0 \\ \max_{s,s'} \left\{ \dfrac{r_a(s')}{r_a(s)} \;\middle|\; s' \in \mathrm{succ}(s) \right\} \text{ otherwise} \end{cases}$$

Note that $\max \emptyset = 0$. Here, $\mathrm{succ}(s)$ denotes the set of immediate successors of the state s as defined by the simple partial order[3]. More precisely, $s' \in \mathrm{succ}(s)$ iff $s' \succ s \wedge \neg \exists s''.\ s' \succ s'' \succ s$. In the case of stochastic ordering, we need to compare the rate functions of two components (the original and the bound), but otherwise the same principle applies:

Definition 18. *The comparative rate measure of PEPA components C and C', where $[\![C]\!]_a = (S, r_a, \boldsymbol{P}_a, L)$ and where $[\![C']\!]_a = (S', r'_a, \boldsymbol{P}'_a, L')$ for action type a, is defined as:*

$$\|C, C'\|_a = \begin{cases} \top \text{ if } \exists s.\ \mathrm{succ}(s) \neq \emptyset \wedge r_a(s) = 0 \\ \max_{s} \left\{ \dfrac{r'_a(s)}{r_a(s)} \right\} \text{ otherwise} \end{cases}$$

In the above definitions, note that the ratio may be undefined (i.e. $r_a(s) = 0$). In this case, we define the ratio to have the value $\top$, which dominates all the reals.

We can now define precisely what we mean by a context, which is slightly different from a conventional process algebra definition, since we care only about those components that can affect the rate at which we perform an activity.

Definition 19. *The context $\mathcal{C}$ of a component C is the set of all components that it can cooperate with, as defined by the system equation. We say that $\mathcal{C}$ is internally bounded by B_{int}, with respect to action type a, if:*

$$\forall C_i \in \mathcal{C},\ \|C_i\|_a \leq B_{int}$$

[3]Note that this successor function comes from our partial order of the state space (i.e. the order that tells us which probabilities we can compare), and not from the transition relation of the Markov chain.

Furthermore, $\mathcal{C}$ and $\mathcal{C}'$ are comparatively bounded by B_{comp}, with respect to action type a, if:

$$\forall C_i \in \mathcal{C}, C_i' \in \mathcal{C}',\ \|C_i, C_i'\|_a \leq B_{comp}$$

Since the internal and comparative bounds depend only on the rate functions, we have a simple algorithm for computing them. If we construct a monotone upper bound of each rate function before bounding the transition matrices, we can compute the internal and comparative bounds of a context as the maximum bounds of the components within the context.

This leads us to the final extension of our definitions — the *context-bounded rate-wise stochastic ordering* and *context-bounded rate-wise monotonicity*, which extend Definitions 13 and 14 respectively. Intuitively, they require the rate function $r_{i,a}$ of component i to not increase faster than the matrices $\boldsymbol{P}_{j,a}$, $j \neq i$, of all its cooperating components allow for.

Definition 20. *Two CTMC components $M_a = (S, r_a, \boldsymbol{P}_a, L)$ and $M_a' = (S, r_a', \boldsymbol{P}_a', L)$, with the same state space S and labelling function L, are ordered $M_a \leq_{\mathrm{rst}}^{B_{comp}} M_a'$ under the* context-bounded rate-wise stochastic ordering*, if:*

1) $\boldsymbol{P}_a \leq_{\mathrm{st}} \boldsymbol{P}_a'$
2) *For all states $s \in S$ such that $r_a(s) \neq r_a'(s)$, if $B_{comp} = \top$ or $r_a(s) = 0$ and $r_a'(s) > 0$, then $\forall s' \prec s.\ \sum_{t \succ s'} \boldsymbol{P}_a'(s,t) = 1$, otherwise:*

$$1 \leq \max\left\{\frac{r_a'(s)}{r_a(s)}, B_{comp}\right\} \leq \min_{s' \prec s}\left\{\frac{1 - \sum_{t \succ s'} \boldsymbol{P}_a(s,t)}{1 - \sum_{t \succ s'} \boldsymbol{P}_a'(s,t)}\right\}$$

We extend the definition of rate-wise monotonicity similarly:

Definition 21. *A CTMC component $M_a = (S, r_a, \boldsymbol{P}_a, L)$ is context-bounded rate-wise monotone with respect to B_{int}, if:*

1) *$\boldsymbol{P}_a$ is monotone.*
2) *For all states $s, s' \in S$ such that $s \prec s'$ and $r_a(s) \neq r_a(s')$, if $B_{int} = \top$ or $r_a(s) = 0$ and $r_a(s') > 0$, then $\forall s'' \prec s.\ \sum_{t \succ s''} \boldsymbol{P}_a(s',t) = 1$, otherwise:*

$$1 \leq \max\left\{\frac{r_a(s')}{r_a(s)}, B_{int}\right\} \leq \min_{s'' \prec s}\left\{\frac{1 - \sum_{t \succ s''} \boldsymbol{P}_a(s,t)}{1 - \sum_{t \succ s''} \boldsymbol{P}_a(s',t)}\right\}$$

Note that this dependence on the context of a component is due to the nature of synchronisation in PEPA.

We can now prove that the CTMC of the system, after composing the individual components, is a monotone, lumpable upper bound of the concrete CTMC, with respect to the ordering on each component. For components C_1 and C_2 with state spaces $(S_1, \prec_1)$ and $(S_2, \prec_2)$ respectively, the $\circledast$ operator preserves stochastic comparison and monotonicity with respect to the lifted orders $(S_1 \times S_2, \prec_1^L)$ and $(S_1 \times S_2, \prec_2^L)$. We say $(s_1, s_2) \prec_1^L (s_1', s_2')$ if $s_1 \prec_1 s_1'$, and $\neg\exists s_2''.\ s_2'' \prec_2 s_2$. We define $\prec_2^L$ similarly.

Theorem 22 (Monotonicity). *Let two PEPA components, C_1 and C_2, occur in contexts $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, where $C_1 \in \mathcal{C}_2$ and $C_2 \in \mathcal{C}_1$. Let $\mathcal{C}_1$ be internally bounded by B_{int}^1 and $\mathcal{C}_2$ by B_{int}^2, for action type a.*

If the CTMC components $[\![C_1]\!]_a = (S_1, r_{1,a}, \boldsymbol{P}_{1,a}L_1)$ and $[\![C_2]\!]_a = (S_2, r_{2,a}, \boldsymbol{P}_{2,a}L_2)$ are context-bounded rate-wise monotone by B_{int}^1 and B_{int}^2 respectively, then $[\![C_1]\!]_a \circledast [\![C_2]\!]_a$ is context-bounded rate-wise monotone by the internal bound B_{int}^3 of the context $\mathcal{C}_1 \cap \mathcal{C}_2$ of $C_1 \underset{\mathcal{L}}{\bowtie} C_2$, for all action sets $\mathcal{L}$.

Theorem 23 (Lumpability). *Let C_1 and C_2 be PEPA models with abstractions $(S_1^\sharp, \alpha_1)$ and $(S_2^\sharp, \alpha_2)$ respectively. Then for all action types a, if $(S_1^\sharp, \alpha_1)$ is a lumpable abstraction of $[\![C_1]\!]_a$ and $(S_2^\sharp, \alpha_2)$ is a lumpable abstraction of $[\![C_2]\!]_a$, then $(S_1^\sharp \times S_2^\sharp, \alpha_1 \times \alpha_2)$ is a lumpable abstraction of $[\![C_1]\!]_a \circledast [\![C_2]\!]_a$.*

Theorem 24 (Stochastic Order). *Consider the PEPA components C_i and C_i', such that $[\![C_i]\!]_a = (S_i, r_{i,a}, \boldsymbol{P}_{i,a}L_i)$ and $[\![C_i']\!]_a = (S_i', r_{i,a}', \boldsymbol{P}_{i,a}'L_i')$ for $i \in \{1,2\}$ and action type a. Let $[\![C_i]\!]_a \leq_{\mathrm{rst}}^{B_{comp}^i} [\![C_i']\!]_a$, with contexts $\mathcal{C}_i \leq_{\mathrm{st}} \mathcal{C}_i'$, where B_{comp}^i is the comparative bound of $\mathcal{C}_i$ and $\mathcal{C}_i'$. If B_{comp}^3 is the comparative bound of the contexts $\mathcal{C}_1 \cap \mathcal{C}_2$ and $\mathcal{C'}_1 \cap \mathcal{C'}_2$, then $[\![C_1]\!]_a \circledast [\![C_2]\!]_a \leq_{\mathrm{rst}}^{B_{comp}^3} [\![C_1']\!]_a \circledast [\![C_2']\!]_a$.*

Monotonicity and stochastic order are preserved due to the definitions we have developed. Lumpability (strong bisimulation) is preserved by the PEPA cooperation combinator [11].

It is straightforward to show that corresponding theorems hold for the addition of CTMC components, if they have the same partially ordered state space. A consequence is that we just need to construct an upper bound for CTMC component of each sequential component for each action type, so that when we expand the $\circledast$ and $\odot$ operators of the Kronecker form, we get an upper bound for the CTMC of the entire model.

V. An Algorithm for Computing a Stochastic Bound of a PEPA Component

Algorithm 1 An algorithm for constructing a context-bounded rate-wise upper-bounding probability transition matrix ($\boldsymbol{P}$, $\mathcal{B}$, and $(S^\sharp, \alpha)$ are taken as inputs)

```
1: R(i,j) ← 0 for all i,j
2: y' ← |S_i^♯|
3: for y ← |S_i^♯| to 1 do
4:    if b(y) = b(B_k) for some B_k then
5:       refresh_sum(P, R, b(y), e(y'))
6:       normalise(R, b(y), e(y'))
7:       for p ← y' to y do
8:          normalise_partition(R, b(p), e(p))
9:       end for
10:      y' ← y − 1
11:   end if
12: end for
```

Our algorithm for constructing an upper bound for a steady state property of a PEPA model is as follows. We take a PEPA

model of the form $C_1 \underset{\mathcal{L}_1}{\bowtie} \cdots \underset{\mathcal{L}_{n-1}}{\bowtie} C_n$. For each action type $a \notin \cup_i \mathcal{L}_i$, we rename a to the internal action type τ — i.e. we group all internal transitions together. For each component C_i, we take as input an abstraction $(S_i^\sharp, \alpha_i)$, and a simple partial order specified by $\mathcal{B}_i = \{\mathcal{B}_{i,1}, \ldots \mathcal{B}_{i,m_i}\}$.

1) For each component C_i, we construct a mapping $\mathcal{I}_i$ from its state space S_i to matrix indices $\{1, \ldots, |S_i|\}$, so that states in the same partition have contiguous indices, which are ordered such that $s \prec s' \Rightarrow \mathcal{I}_i(s) < \mathcal{I}_i(s')$.
2) We compute a lumpable monotone upper-bounding rate function $r'_{i,a}$ from the rate function $r_{i,a}$ of each component C_i and action type $a \in \cup_i \mathcal{L}_i$:

$$r'_{i,a}(s \in \mathcal{B}_{i,k}) = \max \left(\begin{array}{c} \{r_{i,a}(s') \mid \alpha_i(s') = \alpha_i(s)\} \cup \\ \bigcup_{j=k+1}^{m_i} \{r_{i,a}(s') \mid s' \in \mathcal{B}_{i,j}\} \end{array} \right)$$

3) We calculate the internal bound B_{int} and comparative bound B_{comp} for the context of each component and action type $a \in \cup_i \mathcal{L}_i$, using the bounded rate functions.
4) We compute an upper-bounding probability transition matrix $\boldsymbol{R}_{i,a}$ from the transition matrix $\boldsymbol{P}_{i,a}$ of each component C_i and action type $a \in \mathcal{L}$ (Algorithm 1).
5) For the internal action type τ, we uniformise the CTMC component $[\![C_i]\!]_\tau$, and apply Algorithm 1 with no context constraints. Since every state has the same exit rate, the algorithm reduces to that of Fourneau *et al*. over a partially ordered state space.
6) We construct and solve the generator matrix obtained by multiplying out the Kronecker representation of the upper-bounding model[4].

It is important to note that not all of the components in the model necessarily need to have ordering constraints on their state space. For example, if we are interested in a property of just one component — i.e. the projection from the state space of the system onto that of the component — then we have no particular constraints on the probability distributions of the other components. But what does this mean in terms of constructing a bound for that component? The theorems in the previous section only account for when we *need* to bound a component. If we wish to exclude one of the components, we have to assume the 'worst' case — that is to say, that the component does not have any effect on the rest of the system.

Intuitively, when we bound a component, we maximise the probability of moving into higher valued states in the ordering. Since cooperation in PEPA takes the minimum of two rates, it is possible for a component to limit, but not increase, the transition rates for a particular action type. Hence a *monotone* upper bound for a component is a true upper bound in the worst-case context. This means that we can ignore other components and still obtain, locally, an upper bound.

[4]To implement this, we need to explore the transition system generated by the new model, rather than performing the Kronecker multiplications explicitly. This avoids including unreachable states, which would result in a singular generator matrix.

Algorithm 2 *refresh_sum*$(\boldsymbol{P}, \boldsymbol{R}, b, e)$

for $\mathcal{B}_k \leftarrow \mathcal{B}_1$ to $\mathcal{B}_m$ **do**
  $R_{max} \leftarrow \max_{s \in \mathcal{B}_{k-1}} \sum_{j=b}^{|S|} \boldsymbol{R}(s,j)$
  $P_{max} \leftarrow \max_{s \in \mathcal{B}_k} \sum_{j=b}^{|S|} \boldsymbol{P}(s,j)$
  $B_I \leftarrow 1 - \min \left\{ B_{int}, \frac{\max_{s \in \mathcal{B}_k} r(s)}{\max_{s \in \mathcal{B}_k} r'(s)} \right\} (1 - R_{max})$
  $B_C \leftarrow 1 - \min \left\{ B_{comp}, \frac{\max_{s \in \mathcal{B}_{k-1}} r'(s)}{\max_{s \in \mathcal{B}_k} r'(s)} \right\} (1 - P_{max})$
  for $i \leftarrow b(\mathcal{B}_k)$ to $e(\mathcal{B}_k)$ **do**
    if $i \geq b$ **then**
      $\Sigma_{new} \leftarrow \max\{R_{max}, P_{max}, B_I, B_C\}$
    else
      $\Sigma_{new} \leftarrow \max\{R_{max}, P_{max}\}$
    end if
    $P_{new} \leftarrow \Sigma_{new} - \sum_{j'=e+1}^{|S|} \boldsymbol{R}(i,j')$
    $P_{old} \leftarrow \sum_{j=b}^{e} \boldsymbol{P}(i,j)$
    for $j \leftarrow b$ to e **do**
      if $P_{old} > 0$ **then**
        $\boldsymbol{R}(i,j) \leftarrow \frac{\boldsymbol{P}(i,j)}{P_{old}} P_{new}$
      else
        $\boldsymbol{R}(i,j) \leftarrow \frac{1}{e-b+1} P_{new}$
      end if
    end for
  end for
end for

Algorithm 3 *normalise*$(\boldsymbol{R}, b, e)$

for $y \leftarrow 1$ to $|S^\sharp|$ **do**
  $R_{new} \leftarrow \max_{i=b(y)}^{e(y)} \sum_{j=b}^{e} \boldsymbol{R}(i,j)$
  for $i \leftarrow b(y)$ to $e(y)$ **do**
    $R_{old} \leftarrow \sum_{j=b}^{e} \boldsymbol{R}(i,j)$
    for $j \leftarrow b$ to e **do**
      if $R_{old} > 0$ **then**
        $\boldsymbol{R}(i,j) \leftarrow \frac{\boldsymbol{R}(i,j)}{R_{old}} R_{new}$
      else
        $\boldsymbol{R}(i,j) \leftarrow \frac{1}{e-b+1} R_{new}$
      end if
    end for
  end for
end for

Algorithm 4 *normalise_partition*$(\boldsymbol{R}, b, e)$

for $y \leftarrow 1$ to $|S^\sharp|$ **do**
  $R_{average} \leftarrow \frac{1}{e(y)-b(y)+1} \sum_{i=b(y)}^{e(y)} \sum_{j=b}^{e} \boldsymbol{R}(i,j)$
  for $i \leftarrow b(y)$ to $e(y)$ **do**
    for $j \leftarrow b$ to e **do**
      $\boldsymbol{R}(i,j) \leftarrow R_{average}$
    end for
  end for
end for

Let us examine Algorithm 1 in more detail. This takes as input a probability transition matrix $\boldsymbol{P}$, and an empty matrix $\boldsymbol{R}$ (of the same dimensions) in which to construct the monotone and lumpable upper bound. We assume that the upper bound r' of the rate function r has already been constructed, along with the internal and comparative bounds, B_{int} and B_{comp}. We define $b(y)$ and $e(y)$ respectively as the minimum and maximum index in the set $\{\mathcal{I}(s) \mid \mathcal{I}^\sharp(\alpha(s)) = y\}$[5]. $b(\mathcal{B}_k)$ and $e(\mathcal{B}_k)$ are defined similarly for the set $\{\mathcal{I}(s) \mid s \in \mathcal{B}_k\}$.

Algorithm 1 makes use of three sub-procedures:

1) *refresh_sum*$(\boldsymbol{P}, \boldsymbol{R}, b, e)$ (Algorithm 2) ensures that for each ordering block, from indices b to e, the matrix $\boldsymbol{R}$ is context-bounded rate-wise monotone, and an upper bound of $\boldsymbol{P}$. The core of this algorithm is the computation of Σ_{new}, where the bounds B_I and B_C come directly from re-arranging the definitions of context-bounded rate-wise stochastic ordering and monotonicity respectively (Definitions 20 and 21). Note that these additional constraints are only needed for elements on or below the diagonal ($i \geq b$).
   To achieve a new row sum of Σ_{new}, we adjust the individual entries in $\boldsymbol{R}$ so that the relative probabilities are preserved. This choice minimises our impact on the matrix — as the ordering is partial, we can distribute the probability mass within an ordering block in any way.
2) *normalise*$(\boldsymbol{R}, b, e)$ (Algorithm 3) ensures that for each partition in the ordering block from indices b to e, states in the same partition have the same probability of moving to a different ordering block.
3) *normalise_partition*$(\boldsymbol{R}, b, e)$ (Algorithm 4) ensures that each state in the partition from indices b to e has the same probability of moving to another partition. We choose to assign the average transition probabilities.

Essentially, the *normalise* procedure ensures lumpability of ordering blocks — by 'borrowing' probability mass from lower ordering blocks — which preserves monotonicity. *normalise_partition* then ensures lumpability of partitions, by redistributing probability mass within the same ordering block.

Property 25. *The worst-case time complexity for Algorithm 1 is $O(|S|^2)$, where S is the state space of the CTMC component.*

[5] $\mathcal{I}^\sharp$ is defined such that $\mathcal{I}(s) < \mathcal{I}(s') \Rightarrow \mathcal{I}^\sharp(\alpha(s)) \leq \mathcal{I}^\sharp(\alpha(s'))$.

$$
\begin{aligned}
PC_i &= (arrive, \lambda_i).PC'_i + (walkon_{i\oplus 1}, \top).PC_i \\
PC'_i &= (serve_i, \top).PC_i \\
Server_i &= (walkon_{i\oplus 1}, \omega).Server_{i\oplus 1} + (serve_i, \mu).Server'_i \\
Server'_i &= (walk_{i\oplus 1}, \omega).Server_{i\oplus 1}
\end{aligned}
$$

$$
(PC_0 \parallel \cdots \parallel PC_{n-1}) \underset{\{walkon, serve\}}{\bowtie} Server_0
$$

Fig. 3. A PEPA model of a round-robin server

This follows because the *refresh_sum*, *normalise*, and *normalise_partition* procedures each contribute $O(|S|^2)$ operations in the worst case. For each ordering block, we apply *normalise* to the block, and *normalise_partition* to each partition within the block — since the number of states in all the partitions within a block is equal to the number of states within the block, these two procedures result in the same number of operations. Similarly, note that in the *normalise* algorithm, the outermost loop iterates over all the partitions (abstract states), and the second loop iterates over all the states within a partition — this is equivalent to a single loop that iterates over all the concrete states in S.

The complexity of Fourneau's algorithm is also $O(|S|^2)$ [9], therefore compositionality and partial ordering do not affect the worst-case time complexity. Since our algorithm is applied compositionally, however, the state space S is only that of an individual component. This means that overall, the complexity can be much better than directly applying Fourneau's algorithm. If we have N components, each with a state space S, then Fourneau's algorithm has a worst-case time complexity of $O(|S|^{2N})$, whereas our compositional algorithm has complexity $O(N|S|^2A)$, where A is the number of distinct action types in the PEPA model. This excludes the cost of expanding the Kronecker form, but we only expand the lumped Markov chain, which is much smaller than the original.

VI. An Example

We have implemented the stochastic bounding algorithm presented in the previous section as part of the PEPA plug-in for Eclipse [16], and we will now consider its application to a PEPA model. Consider the model in Figure 3. This describes a set of n PCs, which are serviced in a round-robin fashion by a server. Jobs arrive at computer PC_i at rate λ_i, and the service rate of the server is μ. The server in state i moves to state $i \oplus 1 = (i+1) \bmod n$, after serving a job from PC i (with the *walk* activity) or not (with the *walkon* activity).

Table I shows some results for this model when $n = 7$ and $n = 8$, with the rate parameters $\omega = 10$, $\mu = 11$, and $\lambda_i = i + 1$ (we set a different arrival rate for every computer, to prevent the model from being lumpable with respect to our abstraction). The results are shown for two abstractions — one where we aggregate all the $Server'_i$ states, and the other where we also aggregate the $Server_i$ states. In the first case, the abstraction gives an upper bound very close to the actual probability, but the second case (reducing the model to just two states) is too coarse, and gives a poor upper bound. This

Number of PCs	Aggregated States	Original Model			Stochastic Bound Abstraction			ACTMC Abstraction		
		States	Probability	Time (s)	States	Upper Bound	Time (s)	States	Upper Bound	Time (s)
7	None	1792	0.50612	3.1	–	–	–	–	–	–
	$Server'$	–	–	–	8	0.5238	0.1	1024	0.98308	5.9
	$Server, Server'$	–	–	–	2	0.88506	0.0	256	0.81517	1.5
8	None	4096	0.51216	23.8	–	–	–	–	–	–
	$Server'$	–	–	–	9	0.5238	0.1	2304	0.98161	14.7
	$Server, Server'$	–	–	–	2	0.89796	0.0	512	0.84596	3.2

TABLE I
LONG-RUN PROBABILITY OF BEING IN A $Server'_i$ STATE

illustrates the difficulty in choosing a good abstraction, and finding good heuristics is the subject of ongoing work. Note that the durations in the table show the total time taken to both perform the abstraction and solve the CTMC.

We can compare the abstraction in this paper to an alternative based on compositional abstraction of PEPA models to abstract CTMCs [17]. To look at long-run properties, we have implemented a simple value iteration algorithm — this is quite inefficient, and we expect it could perform much better using more sophisticated methods such as those in [20]. We used a direct solver (based on LU-factorisation) for the steady state computations of CTMCs. We can see that stochastic bounds considerably out-performs the ACTMC approach for the first abstraction, but performs worse in the second. In general, we can expect stochastic bounds to perform well when the abstraction is close to being lumpable.

VII. CONCLUSIONS

Stochastic bounding is a powerful technique for reducing the size of a Markov chain, and allows us to obtain bounds on steady state probabilities. We have applied this compositionally to PEPA models, allowing us to bound models where the underlying state space is too large to represent. The algorithm we have presented and implemented enables these bounds to be constructed automatically, given an ordering and partitioning of the state space of each component.

Whilst we have demonstrated the utility of compositional stochastic bounds, there remain many interesting future research directions. Both the choice of the ordering on the state space and its abstraction affect the precision of the bounds that we obtain. Our current approach is to provide a heuristic to choose the ordering (placing the states we are interested in at the top of the ordering), and to provide a graphical interface in our tool [16] to allow the modeller to easily experiment with different abstractions. In the future, it would be interesting to see if these techniques could be combined with counterexample-guided abstraction refinement [10], to automatically find good abstractions for a given property.

In summary, stochastic bounding is a useful method for analysing performance models, and by applying it compositionally to PEPA, we feel that we have widened its application.

ACKNOWLEDGEMENTS

We would like to thank Jane Hillston, Joost-Pieter Katoen, and Perdita Stevens for their many insightful comments and feedback on the work in this paper. This work was supported by a Microsoft Research European Scholarship, and by the Danish Research Council (FTP grant 09-073796).

REFERENCES

[1] The PEPA plug-in for Eclipse. Available to download from: `http://www.dcs.ed.ac.uk/pepa/tools/plugin/`.
[2] O. Abu-Amsha and J.-M. Vincent. An algorithm to bound functionals of Markov chains with large state spaces. In *4th INFORMS Conference on Telecommunications*, 1998.
[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
[4] M. Bernardo. Non-bisimulation-based Markovian behavioral equivalences. *Journal of Logic and Algebraic Programming*, 72(1):3–49, 2007.
[5] D. Daly. *Bounded aggregation techniques to solve large Markov models*. PhD thesis, 2005.
[6] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reduction and refinement strategies for probabilistic analysis. In *Process Algebra and Probabilistic Methods: Performance Modeling and Verification*, volume 2399 of *LNCS*, pages 335–372. Springer, 2002.
[7] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proceedings of LICS '98*, pages 454–465. IEEE Computer Society, 1998.
[8] J.-M. Fourneau and L. Kloul. A precedence PEPA model for performance and reliability analysis. In *Proceedings of EPEW 2006*, volume 4054 of *LNCS*, pages 1–15. Springer, 2006.
[9] J.-M. Fourneau, M. Lecoz, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386:167–185, 2004.
[10] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *Proceedings of CAV '08*, pages 162–175. Springer, 2008.
[11] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
[12] J. Hillston and L. Kloul. An efficient Kronecker representation for PEPA models. In *Proceedings of PAPM-PROBMIV '01*, volume 2165 of *LNCS*, pages 120–135. Springer, 2001.
[13] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *Proceedings of CAV '07*, volume 4590 of *LNCS*, pages 316–329. Springer, 2007.
[14] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer, 1976.
[15] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *SIGMETRICS Performance Evaluation Review*, 13(2):147–154, 1985.
[16] M.J.A. Smith. Abstraction and model checking in the PEPA plug-in for Eclipse. In *Proceedings of QEST '10*, pages 155–156. IEEE, 2010.
[17] M.J.A. Smith. Compositional abstraction of PEPA models for transient analysis. In *Proceedings of EPEW 2010*, volume 6342 of *LNCS*, pages 252–267. Springer, 2010.
[18] D. Stoyan. *Comparison Methods for Queues and Other Stochastic Models*. Wiley & Sons, 1983.
[19] M. Tremolieres, J.M. Vincent, and B. Plateau. Determination of the optimal stochastic upper bound of a Markovian generator. Technical Report Rapport de Recherche RR 906-I, 1992.
[20] R. Wimmer, B. Braitling, B. Becker, E.M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. Theel. Symblicit calculation of long-run averages for concurrent probabilistic systems. In *Proceedings of QEST '10*, pages 27–36. IEEE Computer Society, 2010.

APPENDIX

Theorem 15. *If $M = (S, r, \boldsymbol{P}, L) \leq_{\text{rst}} M' = (S, r', \boldsymbol{P}', L)$ and for all $s \in S$, $r(s) \leq r'(s)$, then $M \leq_{\text{st}} M'$.*

Proof: For any λ that is not exceeded in magnitude by any element of r_a or r'_a, we need to show that $\frac{r_a(\boldsymbol{P}_a - \boldsymbol{I})}{\lambda} + \boldsymbol{I} \leq_{\text{st}} \frac{r'_a(\boldsymbol{P}'_a - \boldsymbol{I})}{\lambda} + \boldsymbol{I}$, by the definition of the stochastic ordering on CTMCs (i.e. applying uniformisation). This corresponds to showing that:

$$\begin{aligned} &\frac{r_a \boldsymbol{P}_a}{\lambda} + \left(1 - \frac{r_a}{\lambda}\right) \boldsymbol{I} \\ \leq_{\text{st}} &\frac{r'_a \boldsymbol{P}'_a}{\lambda} + \left(1 - \frac{r'_a}{\lambda}\right) \boldsymbol{I} \end{aligned}$$

remembering that r_a and r'_a are apparent rate functions, which can be written as vectors. By the definition of the strong stochastic ordering, this requires that, for each row s and for all states s':

$$\begin{aligned} &\frac{r_a(s)}{\lambda} \sum_{t \succ s'} \boldsymbol{P}_a(s,t) + \left(1 - \frac{r_a(s)}{\lambda}\right) \mathbf{1}_{s' \prec s} \\ \leq &\frac{r'_a(s)}{\lambda} \sum_{t \succ s'} \boldsymbol{P}'_a(s,t) + \left(1 - \frac{r'_a(s)}{\lambda}\right) \mathbf{1}_{s' \prec s} \end{aligned}$$

where $\mathbf{1}_{s' \prec s}$ is an indicator value, equal to one if the condition $s' \prec s$ holds, and to zero otherwise. If we are above the diagonal element (i.e. the indicator term evaluates to zero), then the relation holds since $r_a(s) \leq r'_a(s)$ and $\boldsymbol{P}_a \leq_{\text{st}} \boldsymbol{P}'_a$. Otherwise, we have, for $s' \prec s$:

$$\begin{aligned} &\frac{r_a(s)}{\lambda} \sum_{t \succ s'} \boldsymbol{P}_a(s,t) - \frac{r_a(s)}{\lambda} \\ \leq &\frac{r'_a(s)}{\lambda} \sum_{t \succ s'} \boldsymbol{P}'_a(s,t) - \frac{r'_a(s)}{\lambda} \end{aligned}$$

which, on re-arranging, gives:

$$\frac{r'_a(s)}{r_a(s)} \leq \frac{1 - \sum_{t \succ s'} \boldsymbol{P}_a(s,t)}{1 - \sum_{t \succ s'} \boldsymbol{P}'_a(s,t)}$$

But since we know that the left-hand side is less than or equal to the minimum of all possible ratios on the right-hand side, this holds for all s'. ∎

Theorem 16. *If $M = (S, r, \boldsymbol{P}, L)$ is rate-wise monotone, and for all $s \prec s' \in S$, $r(s) \leq r(s')$, then M is monotone.*

Proof: For any λ that is not exceeded in magnitude by any element of r_a, we need to show that $\frac{r_a(\boldsymbol{P}_a - \boldsymbol{I})}{\lambda} + \boldsymbol{I}$ is monotone, by the definition of monotonicity for CTMCs (i.e. applying uniformisation). This corresponds to showing that:

$$\frac{r_a \boldsymbol{P}_a}{\lambda} + \left(1 - \frac{r_a}{\lambda}\right) \boldsymbol{I}$$

is monotone, where we write the apparent rate function r_a as a vector. By the definition of monotonicity, we require for all states s and s', such that $s \prec s'$ (two rows that we compare), and for all states s'' (elements along the row):

$$\begin{aligned} &\frac{r_a(s)}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s,t) + \left(1 - \frac{r_a(s)}{\lambda}\right) \mathbf{1}_{s'' \prec s} \\ \leq &\frac{r_a(s')}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s',t) + \left(1 - \frac{r_a(s')}{\lambda}\right) \mathbf{1}_{s'' \prec s'} \end{aligned}$$

If we are above the diagonal element in both rows (i.e. the indicator terms $\mathbf{1}_{s'' \prec s}$ and $\mathbf{1}_{s'' \prec s'}$ both evaluate to zero), then the relation holds since $r_a(s) \leq r_a(s')$ and $\boldsymbol{P}_a$ is monotone. Otherwise, we have to consider two cases: when $s \prec s'' \prec s'$, and when $s'' \prec s$.

When $s \prec s'' \prec s'$, we have:

$$\begin{aligned} &\frac{r_a(s)}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s,t) \\ \leq &\frac{r_a(s')}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s',t) + 1 - \frac{r_a(s')}{\lambda} \end{aligned}$$

which holds as before, since $1 - \frac{r_a(s')}{\lambda} > 0$.

Finally, when $s'' \prec s$, we have:

$$\begin{aligned} &\frac{r_a(s)}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s,t) - \frac{r_a(s)}{\lambda} \\ \leq &\frac{r_a(s')}{\lambda} \sum_{t \succ s''} \boldsymbol{P}_a(s',t) - \frac{r_a(s')}{\lambda} \end{aligned}$$

which, on re-arranging, gives:

$$\frac{r_a(s')}{r_a(s)} \leq \frac{1 - \sum_{t \succ s''} \boldsymbol{P}_a(s,t)}{1 - \sum_{t \succ s''} \boldsymbol{P}_a(s',t)}$$

But since we know that the left-hand side is less than or equal to the minimum of all possible ratios on the right-hand side, this holds for all s''. ∎

Theorem 22 (Monotonicity). *Let two PEPA components, C_1 and C_2, occur in contexts $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, where $C_1 \in \mathcal{C}_2$ and $C_2 \in \mathcal{C}_1$. Let $\mathcal{C}_1$ be internally bounded by B^1_{int} and $\mathcal{C}_2$ by B^2_{int}, for action type a.*

If the CTMC components $[\![C_1]\!]_a = (S_1, r_{1,a}, \boldsymbol{P}_{1,a}L_1)$ and $[\![C_2]\!]_a = (S_2, r_{2,a}, \boldsymbol{P}_{2,a}L_2)$ are context-bounded rate-wise monotone by B^1_{int} and B^2_{int} respectively, then $[\![C_1]\!]_a \circledast [\![C_2]\!]_a$ is context-bounded rate-wise monotone by the internal bound B^3_{int} of the context $\mathcal{C}_1 \cap \mathcal{C}_2$ of $C_1 \underset{\mathcal{L}}{\bowtie} C_2$, for all action sets $\mathcal{L}$.

To prove this theorem, we will first establish the following two lemmas. Lemma 26 shows that the Kronecker product preserves monotonicity, and Lemma 27 shows that the minimum of two monotone functions is also monotone. We omit the subscript a for clarity, hence we write $\boldsymbol{P}_i$ in place of $\boldsymbol{P}_{i,a}$, and r_i in place of $r_{i,a}$ for $i \in \{1, 2\}$:

Lemma 26. *Let $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ be monotone stochastic matrices describing the PEPA components C_1 and C_2 respectively, which have state spaces $(S_1, \prec_1)$ and $(S_2, \prec_2)$. Then $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2$ is also monotone under the lifted orderings $\prec_1^L$ and $\prec_2^L$ on $S_1 \times S_2$.*

Proof: Consider states $(s_1, s_2) \prec_1^L (s_1', s_2')$, recalling that this implies that $s_1 \prec_1 s_1'$ and $\neg\exists s_2''.\ s_2'' \prec_2 s_2$. We need to show that the following inequality holds, for all $s \in \mathrm{ds}(C_1)$ and $t \in \mathrm{ds}(C_2)$:

$$\sum_{(s',t') \succ_1^L (s,t)} \boldsymbol{P}_1(s_1, s')\boldsymbol{P}_2(s_2, t') \leq \sum_{(s',t') \succ_1^L (s,t)} \boldsymbol{P}_1(s_1', s')\boldsymbol{P}_2(s_2', t')$$

But in order for there to be any states $(s', t') \succ (s, t)$, t must be the smallest state in $(S_2, \prec_2)$. Hence this is equivalent to:

$$\sum_{s' \succ_1 s} \sum_{t' \in S_2} \boldsymbol{P}_1(s_1, s')\boldsymbol{P}_2(s_2, t') \leq \sum_{s' \succ_1 s} \sum_{t' \in S_2} \boldsymbol{P}_1(s_1', s')\boldsymbol{P}_2(s_2', t')$$

which we rewrite to give:

$$\sum_{s' \succ_1 s} \boldsymbol{P}_1(s_1, s') \leq \sum_{s' \succ_1 s} \boldsymbol{P}_1(s_1', s')$$

This holds since $\boldsymbol{P}_1$ is monotone under $(S_1, \prec_1)$. The proof of monotonicity under $(S_1 \times S_2, \prec_2^L)$ follows similarly. ■

Lemma 27. *Let r_1 and r_2 be monotone functions. Then $r_3(s_1, s_2) = \min\{ r_1(s_1), r_2(s_2) \}$ is also monotone, under the orderings $(S_1 \times S_2, \prec_1^L)$ and $(S_1 \times S_2, \prec_2^L)$.*

Proof: Consider states $(s_1, s_2) \prec_1^L (s_1', s_2')$. By definition, $s_1 \prec_1 s_1'$ and either $s_2 \prec_1 s_2'$ or $s_2 = s_2'$. There are two cases to consider:

Case 1: $\min\{ r_1(s_1'), r_2(s_2') \} = r_1(s_1')$. Then:

$$\begin{aligned} \min\{ r_1(s_1), r_2(s_2) \} &\leq r_1(s_1) \\ &\leq r_1(s_1') \\ &\leq \min\{ r_1(s_1'), r_2(s_2') \} \end{aligned}$$

Case 2: $\min\{ r_1(s_1'), r_2(s_2') \} = r_2(s_2')$. Then:

$$\begin{aligned} \min\{ r_1(s_1), r_2(s_2) \} &\leq r_2(s_2) \\ &\leq r_2(s_2') \\ &\leq \min\{ r_1(s_1'), r_2(s_2') \} \end{aligned}$$

Hence r_3 is monotone with respect to $(S_1 \times S_2, \prec_1^L)$. The proof of monotonicity under $(S_1 \times S_2, \prec_2^L)$ follows similarly. ■

Proof: **[Theorem 22]** Let $(S_1, \prec_1)$ and $(S_2, \prec_2)$ be the state spaces of components C_1 and C_2 respectively. We will show that $\min\{ r_1, r_2 \}(\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 - \boldsymbol{I})$ is monotone with respect to $(S_1 \times S_2, \prec_1^L)$.

We know from Lemma 26 that the matrix $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2$ is monotone, and from Lemma 27 that the apparent rate function $\min\{ r_1, r_2 \}$ is monotone increasing. Hence, for all states $(s_1, s_2) \prec_1^L (s_1', s_2')$, we need to show that:

$$\max\left\{ B^3_{int}, \frac{\min\{ r_1(s_1'), r_2(s_2') \}}{\min\{ r_1(s_1), r_2(s_2) \}} \right\} \leq \min_{(t_1,t_2) \prec_1^L (s_1,s_2)} \left\{ \frac{1 - \sum_{(t_1',t_2') \succ_1^L (t_1,t_2)} \boldsymbol{P}_1(s_1, t_1')\boldsymbol{P}_2(s_2, t_2')}{1 - \sum_{(t_1',t_2') \succ_1^L (t_1,t_2)} \boldsymbol{P}_1(s_1', t_1')\boldsymbol{P}_2(s_2', t_2')} \right\}$$

where B^3_{int} is the internal bound of the context $\mathcal{C}''$ of $C \underset{\mathcal{L}}{\bowtie} C'$:

Let (t_1, t_2) be the state under which the ratio on the right hand side is at a minimum. Since $t_1 \prec_1 s_1$ by definition of $\prec_1^L$, we know that the following relation holds:

$$\max\left\{ B^1_{int}, \frac{r_1(s_1')}{r_1(s_1)} \right\} \leq \frac{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1, t_1')}{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1', t_1')}$$

Furthermore, since by definition $B^1_{int} \geq \frac{r_2(s_2')}{r_2(s_2)}$, $B^2_{int} \geq \frac{r_1(s_1')}{r_1(s_1)}$, and $B^3_{int} \leq \min\{B^1_{int}, B^2_{int}\}$, we can infer that:

$$\max\left\{ B^3_{int}, \frac{r_1(s_1')}{r_1(s_1)}, \frac{r_2(s_2')}{r_2(s_2)} \right\} \leq \max\left\{ B^1_{int}, \frac{r_1(s_1')}{r_1(s_1)} \right\}$$

To complete the proof, we need to make use of the following observation:

Observation 28. *For all positive $a, b, c, d \in \mathbb{R}$:*

$$\max\left\{ \frac{a}{b}, \frac{c}{d} \right\} \geq \frac{\min\{ a, c \}}{\min\{ b, d \}}$$

since $\frac{a}{b} \geq \frac{\min\{ a,c \}}{b}$ and $\frac{c}{d} \geq \frac{\min\{ a,c \}}{d}$.

Using this observation, and the fact that t_2 must be the

smallest state in $(S_2, \prec_2)$ by the definition of $\prec_1^L$:

$$
\begin{aligned}
& \max\left\{ B^3_{int}, \frac{\min\{\, r_1(s_1'), r_2(s_2')\,\}}{\min\{\, r_1(s_1), r_2(s_2)\,\}} \right\} \\
\leq\ & \max\left\{ B^3_{int}, \frac{r_1(s_1')}{r_1(s_1)}, \frac{r_2(s_2')}{r_2(s_2)} \right\} \\
\leq\ & \max\left\{ B^1_{int}, \frac{r_1(s_1')}{r_1(s_1)} \right\} \\
& \frac{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1, t_1')}{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1', t_1')} \\
=\ & \frac{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1, t_1') \sum_{t_2' \in S_2} \boldsymbol{P}_2(s_2, t_2')}{1 - \sum_{t_1' \succ_1 t_1} \boldsymbol{P}_1(s_1', t_1') \sum_{t_2' \in S_2} \boldsymbol{P}_2(s_2', t_2')} \\
\leq\ & \frac{1 - \sum_{(t_1', t_2') \succ_1^L (t_1, t_2)} \boldsymbol{P}_1(s_1, t_1') \boldsymbol{P}_2(s_2, t_2')}{1 - \sum_{(t_1', t_2') \succ_1^L (t_1, t_2)} \boldsymbol{P}_1(s_1', t_1') \boldsymbol{P}_2(s_2', t_2')} \\
=\ & \min_{(t_1, t_2) \prec (s_1, s_2)} \left\{ \frac{1 - \sum_{(t_1', t_2') \succ_1^L (t_1, t_2)} \boldsymbol{P}_1(s_1, t_1') \boldsymbol{P}_2(s_2, t_2')}{1 - \sum_{(t_1', t_2') \succ_1^L (t_1, t_2)} \boldsymbol{P}_1(s_1', t_1') \boldsymbol{P}_2(s_2', t_2')} \right\}
\end{aligned}
$$

The proof of monotonicity under $(S_1 \times S_2, \prec_2^L)$ follows similarly.

Thus context-bounded rate-wise monotonicity is preserved by $\circledast$. ■

Theorem 23 (Lumpability). *Let C_1 and C_2 be PEPA models with abstractions $(S_1^\sharp, \alpha_1)$ and $(S_2^\sharp, \alpha_2)$ respectively. Then for all action types a, if $(S_1^\sharp, \alpha_1)$ is a lumpable abstraction of $[\![C_1]\!]_a$ and $(S_2^\sharp, \alpha_2)$ is a lumpable abstraction of $[\![C_2]\!]_a$, then $(S_1^\sharp \times S_2^\sharp, \alpha_1 \times \alpha_2)$ is a lumpable abstraction of $[\![C_1]\!]_a \circledast [\![C_2]\!]_a$.*

Proof: Observe that $[\![C_1]\!]_a \circledast [\![C_2]\!]_a = [\![C]\!]_a$, for $C = C_1 \underset{\{\,a\,\}}{\bowtie} C_2$. Since lumpability and strong bisimilarity are the same, it is clear that $[\![C_1]\!]_a$ is strongly bisimilar to the lumped CTMC component $Abs_{(S_1^\sharp, \alpha_1)}([\![C_1]\!]_a^\sharp)$, and similarly that $[\![C_2]\!]_a$ is strongly bisimilar to $Abs_{(S_2^\sharp, \alpha_2)}([\![C_2]\!]_a^\sharp)$.

It was proven in [11] that the PEPA cooperation combinator preserves strong bisimilarity, and therefore ordinary lumpability, and so it follows that $[\![C]\!]_a$ is strongly bisimilar to:

$$Abs_{(S_1^\sharp, \alpha_1)}([\![C_1]\!]_a) \circledast Abs_{(S_2^\sharp, \alpha_2)}([\![C_2]\!]_a^\sharp) = Abs_{(S^\sharp, \alpha)}([\![C]\!])$$

where $S^\sharp = S_1^\sharp \times S_2^\sharp$ and $\alpha = \alpha_1 \times \alpha_2$. Hence $(S_1^\sharp \times S_2^\sharp, \alpha_1 \times \alpha_2)$ is a lumpable abstraction of $[\![C_1]\!]_a \circledast [\![C_2]\!]_a$. ■

Theorem 24 (Stochastic Order). *Consider the PEPA components C_i and C'_i, such that $[\![C_i]\!]_a = (S_i, r_{i,a}, \boldsymbol{P}_{i,a} L_i)$ and $[\![C'_i]\!]_a = (S'_i, r'_{i,a}, \boldsymbol{P}'_{i,a} L'_i)$ for $i \in \{1,2\}$ and action type a. Let $[\![C_i]\!]_a \leq_{\mathrm{rst}}^{B^i_{comp}} [\![C'_i]\!]_a$, with contexts $\mathcal{C}_i \leq_{\mathrm{st}} \mathcal{C}'_i$, where B^i_{comp} is the comparative bound of $\mathcal{C}_i$ and $\mathcal{C}'_i$. If B^3_{comp} is the comparative bound of the contexts $\mathcal{C}_1 \cap \mathcal{C}_2$ and $\mathcal{C'}_1 \cap \mathcal{C'}_2$, then $[\![C_1]\!]_a \circledast [\![C_2]\!]_a \leq_{\mathrm{rst}}^{B^3_{comp}} [\![C'_1]\!]_a \circledast [\![C'_2]\!]_a$.*

Proof: For clarity, we will omit the subscript a, and hence write $\boldsymbol{P}_i$ in place of $\boldsymbol{P}_{i,a}$, and r_i in place of $r_{i,a}$ for $i \in \{1,2\}$. We omit the proofs that $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \leq_{\mathrm{st}} \boldsymbol{P}'_1 \otimes \boldsymbol{P}'_2$ and that $\min\{r_1, r_2\}(s_1, s_2) \leq \min\{r'_1, r'_2\}(s_1, s_2)$ for all $(s_1, s_2) \in S_1 \times S_2$, since they are very similar to Lemma 26 and Lemma 27 from the proof of Theorem 22 (Monotonicity).

Let $(S_1, \prec_1)$ and $(S_2, \prec_2)$ be the state spaces of components C_1, C'_1 and C_2, C'_2 respectively. We will show that the matrix $\min\{r'_1, r'_2\}(\boldsymbol{P}'_1 \otimes \boldsymbol{P}'_2 - \boldsymbol{I})$ is a context-bounded rate-wise upper bound of $\min\{r_1, r_2\}(\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 - \boldsymbol{I})$, with respect to the ordering $(S_1 \times S_2, \prec_1^L)$.

We need to show that the following inequality holds, for all states (s_1, s_2):

$$\max\left\{ B^3_{comp}, \frac{\min\{r'_1(s_1), r'_2(s_2)\}}{\min\{r_1(s_2), r_2(s_2)\}} \right\} \leq \min_{(t_1,t_2) \prec_1^L (s_1,s_2)} \left\{ \frac{1 - \sum_{(t'_1,t'_2) \succ_1^L (t_1,t_2)} \boldsymbol{P}_1(s_1,t'_1)\boldsymbol{P}_2(s_2,t'_2)}{1 - \sum_{(t'_1,t'_2) \succ_1^L (t_1,t_2)} \boldsymbol{P}'_1(s_1,t'_1)\boldsymbol{P}'_2(s_2,t'_2)} \right\}$$

Let (t_1, t_2) be the state for which the ratio on the right hand side is at a minimum. Since $t_1 \prec_1 s_1$ by definition of $\prec_1^L$, we know that the following relation holds:

$$\max\left\{ B^1_{comp}, \frac{r'_1(s_1)}{r_1(s_1)} \right\} \leq \frac{1 - \sum_{t'_1 \succ_1 t_1} \boldsymbol{P}_1(s_1,t'_1)}{1 - \sum_{t'_1 \succ_1 t_1} \boldsymbol{P}'_1(s_1,t'_1)}$$

Furthermore, since by definition of the comparative bounds, $B^1_{comp} \geq \frac{r'_2(s_2)}{r_2(s_2)}$, and $B^3_{comp} \leq \min\{B^1_{comp}, B^2_{comp}\}$, we can infer that:

$$\max\left\{ B^3_{comp}, \frac{r'_1(s_1)}{r_1(s_1)}, \frac{r'_2(s_2)}{r_2(s_2)} \right\} \leq \max\left\{ B^1_{comp}, \frac{r'_1(s_1)}{r_1(s_1)} \right\}$$

To complete the proof, we make use of Observation 28 from the proof of Theorem 22 (Monotonicity), and the fact that t_2 must be the smallest state in $(S_2, \prec_2)$ by the definition of $\prec_1^L$:

The proof of stochastic ordering under $(S_1 \times S_2, \prec_2^L)$ follows similarly.

Thus the context-bounded rate-wise stochastic ordering is preserved by $\circledast$. ■