

Compositional Abstraction of PEPA Models for Transient Analysis

Michael J. A. Smith^{1*}

Department of Informatics and Mathematical Modelling
Danmarks Tekniske Universitet
Lyngby, Denmark
mjas@imm.dtu.dk

Abstract. Stochastic process algebras such as PEPA allow complex stochastic models to be described in a compositional way, but this leads to state space explosion problems. To combat this, there has been a great deal of work in developing techniques for abstracting Markov chains. In particular, abstract — or interval — Markov chains allow us to aggregate states in such a way as to safely bound transient probabilities of the original Markov chain. Whilst we can apply this technique directly to a PEPA model, it requires us to obtain the CTMC of the model, whose state space may be too large to construct explicitly.

In this paper, we present a compositional application of abstract Markov chains to PEPA, based on a Kronecker representation of the underlying CTMC. This can be used to bound probabilistic reachability properties in the Continuous Stochastic Logic (CSL), and we have implemented this as part of the PEPA plug-in for Eclipse. We conclude with an example application — analysing the performance of a wireless network — and use this to illustrate the impact of the choice of states to aggregate on the precision of the bounds.

1 Introduction

Stochastic modelling is concerned with reasoning about systems with behaviour that evolves over time in a probabilistic manner. It is often natural to describe such systems *compositionally*, which is why stochastic process algebras such as PEPA [7] are widely employed. The problem with compositional formalisms, however, is that their underlying mathematical model (a CTMC in the case of PEPA) might be exponentially larger than its description. This, the *state space explosion problem*, is one of the biggest challenges in adapting analysis techniques to realistically sized models.

To combat this problem, various techniques have been proposed for *abstracting* performance models. The basic idea is to find a smaller model that preserves certain properties of the original. If our model is a Markov chain, one way to approach this is to combine, or *aggregate* certain states in the model — hence reducing its state space. Unfortunately, we cannot do this in general and still end up with a Markov chain, but we can instead introduce non-determinism into the model so that we *bound* the probability of a behaviour happening.

* This work was funded by a Microsoft Research European Scholarship

As an example, say that we have a Markov chain that models a client-server system. The original model might predict a probability of 0.9 of the client receiving a response from the server within one second of sending a request. An *abstraction* of the model might instead give us an *interval* of probabilities — for instance, $[0.8, 0.95]$. Most importantly, we want the abstraction to be *safe*, in that this interval contains the actual probability of the behaviour. The topic of this paper is an approach called *abstract* — or *interval* — *Markov chains* [5, 10], and how we can apply it to PEPA models.

If we want to use this technique to abstract a PEPA model, one way is to generate a CTMC using the semantics of PEPA, and then apply the abstraction to that. The problem with this is that the state space of the model may be too large to explicitly construct — hence we would be unable to construct the abstraction. The idea in this paper is to instead construct the abstraction *compositionally*. That is to say, we abstract each component in a PEPA model individually, and compose these to obtain an abstraction of the entire model. To do so, we use a Kronecker representation for PEPA [8].

The main contributions of this paper are as follows. Firstly, we present an alternative Kronecker representation to that of [8], which avoids the use of functional rates, and prove that this preserves the semantics of PEPA. Secondly, we develop a compositional method for constructing an abstract Markov chain from a PEPA model, and prove that this is a safe abstraction. Finally, we present a small example that illustrates the impact of the choice of states to aggregate on the precision of the bounds obtained. We have implemented the work in this paper as part of the PEPA plug-in for Eclipse [16], which provides a graphical interface for abstracting models, and a model checker for the Continuous Stochastic Logic [2].

A summary of this paper is as follows. We begin in Section 2 by introducing the basic concepts of Markov chains and lumpability, along with the notion of an abstract Markov chain, and the Continuous Stochastic Logic (CSL) for specifying properties. In Section 3 we introduce PEPA along with its Kronecker representation, before showing how to compositionally construct abstract Markov chains from PEPA models in Section 4. Finally, we demonstrate this technique on an example model in Section 5, considering how different abstractions affect the precision of the model checking, before concluding with Section 6. For the proofs of the theorems in this paper, please see <http://lanther.co.uk/papers/EPEW10.pdf>.

2 Markov Chains

Let us begin by formally defining a Markov chain.

Definition 1. A Discrete Time Markov Chain (DTMC) is a tuple (S, \mathbf{P}) , and a Continuous Time Markov Chain (CTMC) is a tuple (S, \mathbf{P}, r) . S is a finite non-empty set of states, $\mathbf{P} : S \times S \rightarrow [0, 1]$ is a stochastic matrix, and $r : S \rightarrow \mathbb{R}_{\geq 0}$ is a function describing the rate of exit for each state. For a CTMC when $r(s) = 0$ — i.e. no transitions are possible from state s — we set $\mathbf{P}(s, s) = 1$, and $\mathbf{P}(s, s') = 0$ for all $s' \neq s$.

The matrix \mathbf{P} describes the probability $\mathbf{P}(s_1, s_2)$ of transitioning between two states s_1 and s_2 of the Markov chain in a single time step. In a DTMC, the duration of this time step is not specified, whereas for a CTMC it is determined by a random variable $X(s)$, such that $\Pr(X(s) \leq t) = 1 - e^{-r(s)t}$ when the state has an exit rate of $r(s)$.

Often, a CTMC is described in terms of its infinitesimal generator matrix \mathbf{Q} , whose elements $\mathbf{Q}(i, j)$ (where $i \neq j$) define the rate of transitioning between states i and j — with diagonal elements given by $\mathbf{Q}(i, i) = -\sum_{j \neq i} \mathbf{Q}(i, j)$. This can be calculated from the rate function r and the probability transition matrix \mathbf{P} as follows:

$$\mathbf{Q} = r(\mathbf{P} - \mathbf{I})$$

Here, we define the multiplication of a matrix \mathbf{M} by a function r as $(r\mathbf{M})(i, j) = r(i)\mathbf{M}(i, j)$. The steady state of an ergodic CTMC with generator matrix \mathbf{Q} is a row vector $\boldsymbol{\pi}$, such that $\boldsymbol{\pi}\mathbf{e} = 1$ (where \mathbf{e} is a column vector of 1s) and $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$.

If a CTMC has a non-zero exit rate for every state, then we obtain its *embedded DTMC* by simply discarding these rates:

Definition 2. The embedded DTMC of $\mathcal{M} = (S, \mathbf{P}, r)$ is $\text{Embed}(\mathcal{M}) = (S, \mathbf{P})$.

This, however, alters the behaviour of the Markov chain by throwing away the relative timing information of its states. In particular, the steady-state solution of the embedded DTMC will be different from that of the CTMC. We can avoid this problem by first *uniformising* the CTMC.

Definition 3. The uniformisation of a CTMC $\mathcal{M} = (S, \mathbf{P}, r)$, with uniformisation rate $\lambda \geq \max_{s \in S} r(s)$ is given by $\text{Unif}_\lambda(\mathcal{M}) = (S, \overline{\mathbf{P}}, \overline{r})$, where $\overline{r}(s) = \lambda$ for all $s \in S$, and:

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

Essentially, uniformisation adjusts the CTMC by inserting self-loops, so that the exit rate of every state is the same.

Consider a Markov chain with a state space S . The basic idea of state space abstraction is to reduce S to an abstract state space S^\sharp , which should be smaller than S . To define an abstraction, we need a mapping between the concrete and abstract states:

Definition 4. An abstraction of a state space S is a pair (S^\sharp, α) , where $\alpha : S \rightarrow S^\sharp$ is a surjective function that maps every concrete state to an abstract state. We define a corresponding concretisation function, $\gamma : S^\sharp \rightarrow \mathcal{P}(S)$, as $\gamma(s^\sharp) = \{s \mid \alpha(s) = s^\sharp\}$.

Aggregating a Markov chain according to an abstraction α means that we do not distinguish between states that map to the same abstract state. Therefore, to still have a Markov chain, the rate of transition between two abstract states must be independent of the particular concrete state we are in. This is called *ordinary lumpability* [13]:

Definition 5. An ordinary lumping of a CTMC $\mathcal{M} = (S, \mathbf{P}, r)$ is an abstraction (S^\sharp, α) such that for all states $s, s' \in S$, if $\alpha(s) = \alpha(s')$ then for all states $s^\sharp \in S^\sharp$:

$$\sum_{t \in \gamma(s^\sharp)} r(s)\mathbf{P}(s, t) = \sum_{t \in \gamma(s^\sharp)} r(s')\mathbf{P}(s', t)$$

An ordinary lumping (S^\sharp, α) induces a new CTMC, in that it completely defines the transition rates between abstract states.

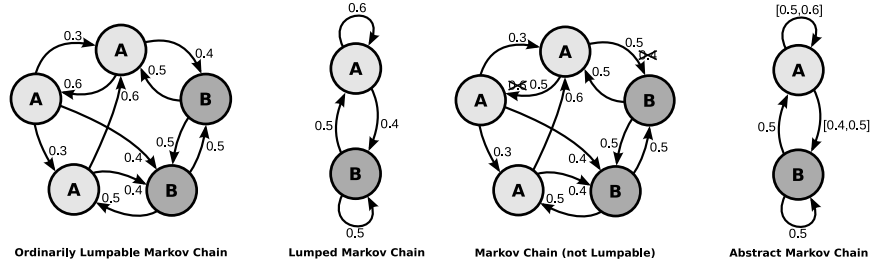


Fig. 1. Ordinary lumpability of a Markov chain

An example of a lumpable Markov chain is shown on the left of Figure 1, which can be viewed as a uniformised CTMC with $\lambda = 1$. Solving this Markov chain, the steady state probability of being in an abstract state (A or B) is equal to the sum of the probabilities of being in each of its constituent states. In other words, solving the aggregated CTMC is equivalent to aggregating the solution of the original CTMC.

2.1 Abstraction of Markov Chains

Unfortunately, it is often the case that a Markov chain we are interested in is not lumpable with respect to a particular abstraction. For example modifying the probabilities slightly in our example, leads to the Markov chain on the right of Figure 1. We can still construct an abstraction, however, if we label transitions with an *interval* of probabilities, or rates — this is called an *abstract Markov chain*. The notion of an abstract DTMC was introduced in [5, 10], and extended to continuous time in [12] by means of uniformisation. The idea is closely related to Markov Decision Processes (MDPs) [14], in that transitions have both a probabilistic and a non-deterministic component.

An abstract CTMC (ACTMC) is defined as follows (as per [12]):

Definition 6. An ACTMC is a tuple $(S^\#, \mathbf{P}^L, \mathbf{P}^U, \lambda, L)$, where $S^\#$ is a finite non-empty set of states, and $\mathbf{P}^L, \mathbf{P}^U : S^\# \times S^\# \rightarrow [0, 1]$ are sub-stochastic and super-stochastic matrices respectively, such that for all states $s, s' \in S^\#, \mathbf{P}^L(s, s') \leq \mathbf{P}^U(s, s')$. λ is the uniformisation constant, denoting the exit rate for every state, and we have a labelling function $L : S^\# \times AP \rightarrow \{\text{tt}, \text{ff}, ?\}$.

Note that the labelling function contains a third truth assignment, ‘?’, which signifies uncertainty (some of the concrete states satisfy the property, but some do not). The truth assignments naturally form a partial order relating to the information they provide: $\text{ff} \sqsubseteq ?$ and $\text{tt} \sqsubseteq ?$, and $\neg ? = ?$.

This definition of an ACTMC induces a natural partial order.

Definition 7. If $\mathcal{M}_1^\# = (S_1^\#, \mathbf{P}_1^L, \mathbf{P}_1^U, \lambda_1, L_1)$ and $\mathcal{M}_2^\# = (S_2^\#, \mathbf{P}_2^L, \mathbf{P}_2^U, \lambda_2, L_2)$, then we say that $\mathcal{M}_1^\# \leq \mathcal{M}_2^\#$ if:

1. $S_1^\# = S_2^\#, \lambda_1 = \lambda_2$ and $L_1 = L_2$.
2. For all $s, s' \in S_1^\#, \mathbf{P}_2^L(s, s') \leq \mathbf{P}_1^L(s, s') \leq \mathbf{P}_1^U(s, s') \leq \mathbf{P}_2^U(s, s')$

Intuitively, \mathcal{M}_2^\sharp is an over-approximation of \mathcal{M}_1^\sharp , since a greater range of transition probabilities are possible.

If we have a uniform CTMC \mathcal{M} and an abstraction (S^\sharp, α) , then we can uniquely define an ACTMC (the closest abstraction) as follows:

Definition 8. The ACTMC $\mathcal{M}^\sharp = \text{Abs}_{(S^\sharp, \alpha)}(\mathcal{M})$ induced by an abstraction (S^\sharp, α) on a uniform CTMC $\mathcal{M} = (S, \mathbf{P}, r, L)$ is defined as follows. Since \mathcal{M} is uniformised, there is a constant λ such that $r(s) = \lambda$ for all $s \in S$:

$$\text{Abs}_{(S^\sharp, \alpha)}(\mathcal{M}) = (S^\sharp, \mathbf{P}^L, \mathbf{P}^U, \lambda, L^\sharp)$$

where:

$$\begin{aligned} \mathbf{P}^L(s_1^\sharp, s_2^\sharp) &= \min_{s_1 \in \gamma(s_1^\sharp)} \sum_{s_2 \in \gamma(s_2^\sharp)} \mathbf{P}(s_1, s_2) & \mathbf{P}^U(s_1^\sharp, s_2^\sharp) &= \max_{s_1 \in \gamma(s_1^\sharp)} \sum_{s_2 \in \gamma(s_2^\sharp)} \mathbf{P}(s_1, s_2) \\ L^\sharp(s^\sharp, a) &= \begin{cases} \mathbf{tt} & \text{if } \forall s \in \gamma(s^\sharp). L(s, a) = \mathbf{tt} \\ \mathbf{ff} & \text{if } \forall s \in \gamma(s^\sharp). L(s, a) = \mathbf{ff} \\ ? & \text{otherwise} \end{cases} \end{aligned}$$

2.2 Continuous Stochastic Logic (CSL)

To describe properties of a Markov chain, it is useful to have a logic for expressing them. Continuous Stochastic Logic (CSL) [2] is a branching-time temporal logic that is widely used for reasoning about CTMCs. In particular, it allows us to talk about the *probability* of a state satisfying some temporal property, and the *time interval* in which the property must hold.

Formulae in CSL are classified into *state formulae* Φ , and *path formulae* φ . The former are properties of individual states in the CTMC — for example, that the state corresponds to an error. The latter are properties that hold of paths (sequences of states) through the CTMC — for example, that no errors occur before we reach a goal state.

The syntax of CSL is as follows, for $\trianglelefteq \in \{\leq, \geq\}$, $a \in AP$ and $p \in [0, 1]$, and where I is a non-empty interval over $\mathbb{R}_{\geq 0} \cup \{\infty\}$:

$$\begin{aligned} \Phi &::= \mathbf{tt} \mid a \mid \Phi \wedge \Phi \mid \neg \Phi \mid \mathcal{S}_{\trianglelefteq p}(\Phi) \mid \mathcal{P}_{\trianglelefteq p}(\varphi) \\ \varphi &::= X^I \Phi \mid \Phi \mathcal{U}^I \Phi \end{aligned}$$

Aside from atomic propositions $a \in AP$ and the standard logical connectives, there are three types of property we can express in CSL:

- A *steady state property* — $\mathcal{S}_{\trianglelefteq p}(\Phi)$ is satisfied if the steady state probability of being in the set of states satisfying Φ is $\trianglelefteq p$.
- A *timed next property* — $\mathcal{P}_{\trianglelefteq p}(X^I \Phi)$ is satisfied of a state s if the probability that we leave the state at time $t \in I$, and the next state satisfies Φ , is $\trianglelefteq p$.
- A *timed until property* — $\mathcal{P}_{\trianglelefteq p}(\Phi_1 \mathcal{U}^I \Phi_2)$ is satisfied of a state s if the probability that we reach a state that satisfies Φ_2 at a time $t \in I$, and we only pass through states that satisfy Φ_1 along the way, is $\trianglelefteq p$.

In the model checker PRISM [9], we can also write *quantitative properties*: $\mathcal{S}_{=?}(\Phi)$ and $\mathcal{P}_{=?}(\varphi)$. These allow us to calculate, rather than just compare, probabilities.

The focus in this paper will be on bounding the probability of *timed until properties*. As an example of such a property, consider $\mathcal{P}_{\geq 0.9}(\neg \text{Error } U^{[0,10]} \text{Completed})$, where $AP = \{ \text{Error}, \text{Completed} \}$. This is satisfied by all states from which there is a probability of at least 0.9 of reaching a ‘Completed’ state within 10 time units, without encountering any ‘Error’ states before then. In [12], a three-valued semantics of CSL is given, along with a model checking algorithm for timed until properties.

3 The Performance Evaluation Process Algebra

So far we have looked only at bounding Markov chains, but the aim of this paper is to present a compositional approach to bounding stochastic process algebra models. To this end, let us introduce the Performance Evaluation Process Algebra (PEPA) [7] — a compositional formalism with CTMC semantics. In PEPA, a *system* is a set of concurrent *components*, which are capable of performing *activities*. An activity $a \in \mathcal{Act}$ is a pair (a, r) , where $a \in \mathcal{A}$ is its action type, and $r \in \mathbb{R}_{\geq 0} \cup \{\top\}$ is the rate of the activity. This rate parameterises an exponential distribution, and if unspecified (denoted \top), the activity is said to be *passive*. In this case, another component is needed to actively drive the rate of this action. PEPA terms have the following syntax:

$$\begin{aligned} C_S &:= (a, r).C_S \mid C_S + C_S \mid A \\ C_M &:= C_S \mid C_M \underset{L}{\bowtie} C_M \mid C_M/L \end{aligned} \quad (1)$$

We call a term C_S a *sequential component*, and a term C_M a *model component*. To define a PEPA model, we need to identify a particular model component that describes its initial configuration, which we call the *system equation*. The meaning of each combinator is as follows:

- *Prefix* $((a, r).C)$: the component can carry out an activity of type a at rate r to become the component C .
- *Choice* $(C_1 + C_2)$: the system may behave either as component C_1 or C_2 . The current activities of both components are enabled, and the first activity to complete determines which component proceeds. The other component is discarded.
- *Cooperation* $(C_1 \underset{L}{\bowtie} C_2)$: the components C_1 and C_2 synchronise over the cooperation set L . For activities whose type is not in L , the two components proceed independently. Otherwise, they must perform the activity together, at the rate of the slowest component.
- *Hiding* (C/L) : the component behaves as C , except that activities whose type is in L are hidden, and appear externally as the unknown type τ .
- *Constant* $(A \stackrel{\text{def}}{=} C)$: component C has the name A .

The operational semantics of PEPA defines a labelled multi-transition system, which induces a *derivation graph* for a given component. Since the duration of a transition in this graph is given by an exponentially distributed random variable, this corresponds to a CTMC. An example PEPA model with two components is shown in Figure 2.

To apply a compositional abstraction to a PEPA model, we need to consider the structure of its underlying CTMC. It was shown in [8] how the generator matrix of this Markov chain can be represented in a compositional, *Kronecker* form. Here, we

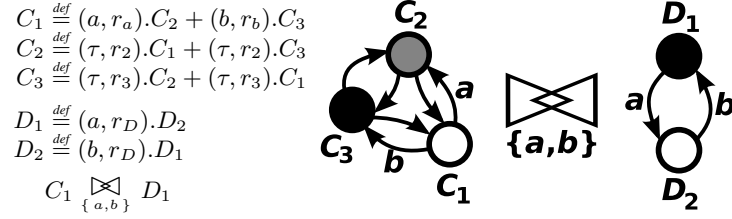


Fig. 2. An example PEPA model and its graphical representation

develop a slight variation to avoid the need for functional rates. If we consider the system equation of a PEPA model, it has the following form:

$$C_1 \boxtimes_{L_1} \cdots \boxtimes_{L_{N-1}} C_N \quad (2)$$

We can ignore the hiding operator C/L without loss of generality, since we can always rename action types to avoid name conflicts between components.

The semantics of PEPA allows us to induce a CTMC from the system equation of a PEPA model. If we look at a *fragment* of the system equation, we can also induce a CTMC following the PEPA semantics — but only if the fragment cannot perform any passive activities. In order to describe the behaviour of a fragment that *can* perform passive activities, we will generalise the notion of a generator matrix. In particular, if we consider a sequential component C_i , having a state space $S_i = \text{ds}(C_i)$, we can write a ‘partial’ generator matrix for the component as follows:

$$Q_i = \sum_{a \in \text{Act}(C_i)} Q_{i,a} = \sum_{a \in \text{Act}(C_i)} r_{i,a} (\mathbf{P}_{i,a} - \mathbf{I}_{|S_i|}) \quad (3)$$

Here, each $Q_{i,a}$ is an $|S_i| \times |S_i|$ matrix that describes the behaviour of C_i due to activities of type a . Importantly, the elements of $Q_{i,a}$ come from the set $\mathbb{R} \cup (\mathbb{R} \times \{\top\})$ — i.e. they correspond to either an active rate (in \mathbb{R}), or a passive rate (in $\mathbb{R} \times \{\top\}$). We define addition and multiplication over these elements as follows, for $r, s \in \mathbb{R}$:

$$\begin{array}{c|cc}
+ & s & (s, \top) \\
\hline
r & r+s & r \\
(r, \top) & s & (r+s, \top)
\end{array}
\quad
\begin{array}{c|cc}
\times & s & (s, \top) \\
\hline
r & rs & (rs, \top) \\
(r, \top) & (rs, \top) & (rs, \top)
\end{array}$$

We further decompose each $Q_{i,a}$ into a rate function $r_{i,a}$ and a probability transition matrix $\mathbf{P}_{i,a}$ — $r_{i,a} : S_i \rightarrow \mathbb{R}_{\geq 0} \cup \{\top\}$ gives the rate of action type a for each state in S_i , $\mathbf{P}_{i,a}$ gives the next-state transition probabilities conditional on performing an activity of type a , and $\mathbf{I}_{|S_i|}$ is the $|S_i| \times |S_i|$ identity matrix. If, for a state s , $r_{i,a}(s) = 0$, we write $\mathbf{P}_{i,a}(s, s) = 1$ and $\mathbf{P}_{i,a}(s, s') = 0$ for $s' \neq s$. Since the rate is zero, we could effectively have chosen any values for this row, but this choice encodes the fact that we remain in the same state.

To build a compositional representation of the generator matrix Q of an arbitrary PEPA model, whose system equation is structured as in Equation 2, we need to combine the individual generator matrices $Q_{i,a}$ in an appropriate way. More precisely, the

compositional representation of \mathbf{Q} has to describe the same CTMC as induced by the semantics of the PEPA model. Because cooperation between two PEPA components uses the *minimum* of two rates, we need to be especially careful that this leads to the correct apparent rate for each state and action type.

To do this, a Kronecker representation for PEPA was developed in [8], using functional rates. We take a slightly different approach here, in that we ensure that functional rates depend only on the state of a single component, at the expense of having more complicated combinators for combining the $\mathbf{Q}_{i,a}$ matrices. This leads to a representation that is a little less elegant mathematically, but which enables us to more easily establish and prove the results in this paper. To describe the generator matrix term $\mathbf{Q}_{i,a}$ for activities of type a in a component C_i , we will use the shorthand $(r_{i,a}, \mathbf{P}_{i,a})$, which is defined as follows:

$$(r_{i,a}, \mathbf{P}_{i,a}) = r_{i,a} (\mathbf{P}_{i,a} - \mathbf{I}_{|S_i|}) = \mathbf{Q}_{i,a}$$

where S_i is the state space of C_i .

Recall that $r_{i,a}$ is an apparent rate function (depending only on the state of C_i) and $\mathbf{P}_{i,a}$ is a probabilistic transition matrix, as in Equation 3. If a component C_i cannot perform any activities of action type a , we define its generator matrix term to be $\mathbf{Q}_{i,a} = (r_{\perp}, \mathbf{I}_{|S_i|})$, where $r_{\perp}(s) = 0$ for all $s \in S_i$.

Using this notation, we can now introduce two Kronecker operators, \otimes and \odot , which correspond to cooperating and independent activities. If two components C_1 and C_2 cooperate over an action type a , we will use the operator \otimes , which is defined as:

$$(r_{1,a}, \mathbf{P}_{1,a}) \otimes (r_{2,a}, \mathbf{P}_{2,a}) = (\min\{r_{1,a}, r_{2,a}\}, \mathbf{P}_{1,a} \otimes \mathbf{P}_{2,a}) \quad (4)$$

where $\min\{r_{1,a}, r_{2,a}\}(s_1, s_2) = \min\{r_{1,a}(s_1), r_{2,a}(s_2)\}$ for all $s_1 \in S_1$ and $s_2 \in S_2$. The operator \otimes denotes the Kronecker product of two matrices.

If, on the other hand, C_1 and C_2 independently perform activities of type a , we will use the operator \odot , which we define in terms of \otimes :

$$(r_{1,a}, \mathbf{P}_{1,a}) \odot (r_{2,a}, \mathbf{P}_{2,a}) = (r_{1,a}, \mathbf{P}_{1,a}) \otimes (r_{\top}, \mathbf{I}_{|S_2|}) + (r_{\top}, \mathbf{I}_{|S_1|}) \otimes (r_{2,a}, \mathbf{P}_{2,a}) \quad (5)$$

where $r_{\top}(s) = \top$ for all s . This is intuitively the Kronecker sum defined over our (r, \mathbf{P}) notation. Here, the $+$ operator is normal matrix addition at the level of the generator matrices, but to continue to use our (r, \mathbf{P}) representation we will define it compositionally:

Theorem 1. *Consider two generator matrices $\mathbf{Q}_1 = (r_1, \mathbf{P}_1)$ and $\mathbf{Q}_2 = (r_2, \mathbf{P}_2)$, corresponding to the same state space S — \mathbf{Q}_1 and \mathbf{Q}_2 are both $|S| \times |S|$ matrices. Then $\mathbf{Q}_1 + \mathbf{Q}_2$ can be written as follows:*

$$\mathbf{Q}_1 + \mathbf{Q}_2 = (r_1, \mathbf{P}_1) + (r_2, \mathbf{P}_2) = \left(r_1 + r_2, \frac{r_1}{r_1 + r_2} \mathbf{P}_1 + \frac{r_2}{r_1 + r_2} \mathbf{P}_2 \right)$$

where $(r_1 + r_2)(s) = r_1(s) + r_2(s)$, 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 $s \in S$.

The coefficients of P_1 and P_2 describe the relative probability of taking a transition corresponding to Q_1 or Q_2 . Note that they are functions, in that each row of the matrix is multiplied by a different value — this is because the relative apparent rate can differ between states.

For both of our Kronecker operators, \otimes and \odot , the resulting generator matrix term is for the component $C_1 \underset{L}{\boxtimes} C_2$, and has a state space of $S_1 \times S_2$. This Cartesian state space does not in general correspond to the derivative set $\text{ds}(C_1 \underset{L}{\boxtimes} C_2)$, since it may contain unreachable states. In practice, however, we never expand out the Kronecker form directly, in the sense of performing the tensor multiplications — after using the tensor representation to perform the abstraction, we generate only its *reachable* state space for the purposes of model checking.

We can now define our Kronecker representation for PEPA models, using the \otimes and \odot operators.

Definition 9. Given a PEPA model $C = C_1 \underset{L_1}{\boxtimes} \cdots \underset{L_{N-1}}{\boxtimes} C_N$, its Kronecker form $Q(C)$ is defined as follows:

$$Q(C_1 \underset{L_1}{\boxtimes} \cdots \underset{L_{N-1}}{\boxtimes} C_N) = \sum_{a \in \text{Act}(C)} Q_a(C_1 \underset{L_1}{\boxtimes} \cdots \underset{L_{N-1}}{\boxtimes} C_N)$$

where $\text{Act}(C)$ is the set of all action types that occur in C (both synchronised and independent), and Q_a is defined inductively as follows:

$$\begin{aligned} Q_a(C_i) &= (r_{i,a}, P_{i,a}) && \text{if } C_i \text{ is a sequential component} \\ Q_a(C_i \underset{L}{\boxtimes} C_j) &= \begin{cases} Q_a(C_i) \otimes Q_a(C_j) & \text{if } a \in L \\ Q_a(C_i) \odot Q_a(C_j) & \text{if } a \notin L \end{cases} \end{aligned}$$

The following theorem establishes the correctness of our Kronecker representation, in that it defines an equivalent CTMC to that induced by the PEPA semantics:

Theorem 2. For all well-formed¹ PEPA models C , the CTMC induced by the semantics of PEPA and the CTMC described by the generator matrix $Q(C)$, projected onto the derivative set $\text{ds}(C)$ (the reachable state space of C), are isomorphic.

As an example of how the Kronecker form is applied, let us take the PEPA model from Figure 2. Here, there are two sequential components (C_1 and D_1) and three action types — we cooperate over a and b , but τ is performed independently. Applying our Kronecker form, we arrive at the following structure for $Q(C_1 \underset{\{a,b\}}{\boxtimes} D_1)$:

$$\begin{aligned} Q(C_1 \underset{\{a,b\}}{\boxtimes} D_1) &= Q_\tau(C_1) \odot Q_\tau(D_1) \\ &\quad + Q_a(C_1) \otimes Q_a(D_1) \\ &\quad + Q_b(C_1) \otimes Q_b(D_1) \end{aligned}$$

¹ A well-formed PEPA model is one in which cooperation occurs only at the level of the system equation. If a model has a single system equation, the PEPA syntax given in Equation 1 implicitly guarantees that it is well-formed.

If we had an additional copy of component D , such that the system equation was $C_1 \boxtimes_{\{a,b\}} (D_1 \parallel D_1)$, then $\mathbf{Q}(C_1 \boxtimes_{\{a,b\}} (D_1 \parallel D_1))$ would be written as:

$$\begin{aligned} \mathbf{Q}(C_1 \boxtimes_{\{a,b\}} (D_1 \parallel D_1)) &= \mathbf{Q}_\tau(C_1) \odot (\mathbf{Q}_\tau(D_1) \odot \mathbf{Q}_\tau(D_1)) \\ &\quad + \mathbf{Q}_a(C_1) \otimes (\mathbf{Q}_a(D_1) \odot \mathbf{Q}_a(D_1)) \\ &\quad + \mathbf{Q}_b(C_1) \otimes (\mathbf{Q}_b(D_1) \odot \mathbf{Q}_b(D_1)) \end{aligned}$$

Returning to our model with just two components, let us consider the internal action type τ of component C . We can write the corresponding generator matrix term, $\mathbf{Q}_\tau(C_1) = (r_{C,\tau}, \mathbf{P}_{C,\tau})$ as follows:

$$\mathbf{Q}_\tau(C_1) = \left(\begin{bmatrix} 0 \\ 2r_2 \\ 2r_3 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 2r_2 \\ 2r_3 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)$$

Although it has been written as a vector in the above, it is important to remember that the rate function is a *function*, and is interpreted as multiplying each row of the probability transition matrix by the corresponding rate. The generator matrix for the entire model can be written in its Kronecker form as follows, where we expand out the \otimes and \odot operators to show the tensor products \otimes :

$$\begin{aligned} \mathbf{Q} &= \min \left\{ \begin{bmatrix} 0 \\ 2r_2 \\ 2r_3 \end{bmatrix}, \begin{bmatrix} \top \\ \top \\ \top \end{bmatrix} \right\} \left(\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ &\quad + \min \left\{ \begin{bmatrix} \top \\ \top \\ \top \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ &\quad + \min \left\{ \begin{bmatrix} r_a \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} r_D \\ 0 \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ &\quad + \min \left\{ \begin{bmatrix} r_b \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ r_D \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{aligned} \tag{6}$$

Note that the second term in the above evaluates to zero, because the D component does not perform any internal τ activities.

4 Compositional Abstraction of PEPA Models

To model check transient CSL properties, excluding the timed next operator, we will show in this section how to compositionally construct an ACTMC from the Kronecker representation of a PEPA model. We will begin by defining an *ACTMC component*, in which we bound the probability transition matrix and the rate function separately.

Definition 10. An ACTMC component is a tuple $(S^\sharp, \mathbf{P}^L, \mathbf{P}^U, r^L, r^U, L^\sharp)$, where S^\sharp , \mathbf{P}^L , \mathbf{P}^U and L^\sharp are defined as per an ACTMC, and the rate functions $r^L, r^U : S^\sharp \rightarrow \mathbb{R}_{\geq 0}$ satisfy $r^L(s) \leq r^U(s)$ for all $s \in S^\sharp$.

An ACTMC component induces an ACTMC as follows:

Definition 11. Let $\mathcal{M}^{\#} = (S^{\#}, \mathbf{P}_a^L, \mathbf{P}_a^U, r_a^L, r_a^U, L^{\#})$ be an ACTMC component. We can define an ACTMC with uniformisation constant $\lambda \geq \max_{s \in S^{\#}} r_a^U(s)$ as:

$$ACTMC_{\lambda}(\mathcal{M}^{\#}) = (S^{\#}, \mathbf{P}^L, \mathbf{P}^U, \lambda, L^{\#})$$

where \mathbf{P}^L and \mathbf{P}^U are defined as follows:

$$\mathbf{P}^L(s, s') = \begin{cases} \frac{r_a^L(s)}{\lambda} \mathbf{P}_a^L(s, s') & \text{if } s \neq s' \\ \frac{r_a^L(s)}{\lambda} \mathbf{P}_a^L(s, s) + \left(1 - \frac{r_a^U(s)}{\lambda}\right) & \text{otherwise} \end{cases}$$

$$\mathbf{P}^U(s, s') = \begin{cases} \frac{r_a^U(s)}{\lambda} \mathbf{P}_a^U(s, s') & \text{if } s \neq s' \\ \frac{r_a^U(s)}{\lambda} \mathbf{P}_a^U(s, s) + \left(1 - \frac{r_a^L(s)}{\lambda}\right) & \text{otherwise} \end{cases}$$

The intuition here is that we add the diagonal elements to account for the term $1 - \frac{r_a}{\lambda} \mathbf{I}$ that appears in the uniformised probabilistic transition matrix:

$$\mathbf{P} = \frac{1}{\lambda} \mathbf{Q} + \mathbf{I} = \frac{1}{\lambda} r_a (\mathbf{P}_a - \mathbf{I}) + \mathbf{I}$$

Since we only have upper and lower bounds for the rates r_a , we need to choose the most conservative values to ensure that the bound is correct. This comes at a loss of precision, but this is necessary if we are to combine the ACTMC components and still end up with a safe ACTMC — with respect to the ACTMC obtained from the Markov chain of the PEPA model. In this context, an abstract CTMC $\mathcal{M}_2^{\#}$ is a safe approximation of $\mathcal{M}_1^{\#}$ if $\mathcal{M}_1^{\#} \leq \mathcal{M}_2^{\#}$, as per Definition 7.

Given a sequential PEPA component C_i with state space S_i , we can define a CTMC $\mathcal{M}_{i,a} = (S_i, \mathbf{P}_{i,a}, r_{i,a}, L_i)$ describing the behaviour of the component with respect to action type a . This will not necessarily be ergodic, since some states of C_i might not perform an action of type a . The component $\mathbf{Q}_{i,a}$ corresponding to $\mathcal{M}_{i,a}$ in the Kronecker representation of the PEPA model is defined as $\mathbf{Q}_{i,a} = r_{i,a}(\mathbf{P}_{i,a} - \mathbf{I})$. From the CTMC $\mathcal{M}_{i,a}$, given an abstraction $(S^{\#}, \alpha)$, we can derive an ACTMC component:

Definition 12. The ACTMC component induced by an abstraction $(S^{\#}, \alpha)$ on a CTMC $\mathcal{M} = (S, \mathbf{P}, r, L)$ is defined as:

$$AbsComp_{(S^{\#}, \alpha)}(\mathcal{M}) = (S^{\#}, \mathbf{P}^L, \mathbf{P}^U, r^L, r^U, L^{\#})$$

where:

$$\mathbf{P}^L(s_1^{\#}, s_2^{\#}) = \min_{s_1 \in \gamma(s_1^{\#})} \sum_{s_2 \in \gamma(s_2^{\#})} \mathbf{P}(s_1, s_2) \quad r^L(s^{\#}) = \min_{s \in \gamma(s^{\#})} r(s)$$

$$\mathbf{P}^U(s_1^{\#}, s_2^{\#}) = \max_{s_1 \in \gamma(s_1^{\#})} \sum_{s_2 \in \gamma(s_2^{\#})} \mathbf{P}(s_1, s_2) \quad r^U(s^{\#}) = \max_{s \in \gamma(s^{\#})} r(s)$$

$$L^{\#}(s^{\#}, a) = \begin{cases} \mathbf{tt} & \text{if } \forall s \in \gamma(s^{\#}). L(s, a) = \mathbf{tt} \\ \mathbf{ff} & \text{if } \forall s \in \gamma(s^{\#}). L(s, a) = \mathbf{ff} \\ ? & \text{otherwise} \end{cases}$$

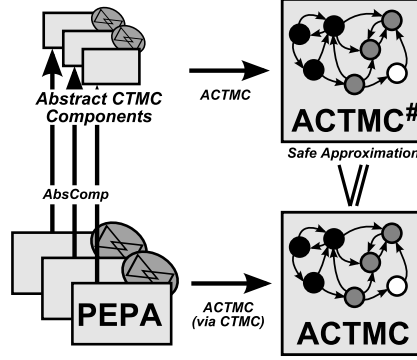


Fig. 3. Safety property of Abstract CTMC Components (Theorem 4)

Theorem 3. Consider a CTMC $\mathcal{M} = (S, \mathbf{P}, r, L)$. For any uniformisation constant $\lambda \geq \max_{s \in S} r(s)$, and any abstraction (S^\sharp, α) on \mathcal{M} , the following holds:

$$\text{Abs}_{(S^\sharp, \alpha)}(\text{Unif}_\lambda(\mathcal{M})) \leq \text{ACTMC}_\lambda(\text{AbsComp}_{(S^\sharp, \alpha)}(\mathcal{M}))$$

This theorem states that ACTMC components *safely approximate* ACTMCs. In other words, an ACTMC component gives an over-approximation of the probability transition intervals, compared to directly generating an ACTMC.

Consider two ACTMC components, $\mathcal{M}_{1,a}^\sharp = (S_1^\sharp, \mathbf{P}_{1,a}^L, \mathbf{P}_{1,a}^U, r_{1,a}^L, r_{1,a}^U, L_1^\sharp)$ and $\mathcal{M}_{2,a}^\sharp = (S_2^\sharp, \mathbf{P}_{2,a}^L, \mathbf{P}_{2,a}^U, r_{2,a}^L, r_{2,a}^U, L_2^\sharp)$. We can construct a new ACTMC component, corresponding to the two components cooperating over action type a as follows:

$$\begin{aligned} \mathcal{M}_{1,a}^\sharp \otimes \mathcal{M}_{2,a}^\sharp = \\ \left(S_1^\sharp \times S_2^\sharp, \mathbf{P}_{1,a}^L \otimes \mathbf{P}_{2,a}^L, \mathbf{P}_{1,a}^U \otimes \mathbf{P}_{2,a}^U, \min\{r_{1,a}^L, r_{2,a}^L\}, \min\{r_{1,a}^U, r_{2,a}^U\}, L_1^\sharp \times L_2^\sharp \right) \end{aligned}$$

where the new labelling function is $L_1^\sharp \times L_2^\sharp((s_1, s_2), a) = L_1^\sharp(s_1, a) \wedge L_2^\sharp(s_2, a)$. The minimum operators in the above come from the semantics of cooperation in PEPA, so they apply to both the upper and lower bounds for the rates. If the two components do not cooperate over the action type a (i.e. they perform activities of type a independently), then the new ACTMC component will instead be:

$$\mathcal{M}_{1,a}^\sharp \odot \mathcal{M}_{2,a}^\sharp = \left(S_1^\sharp \times S_2^\sharp, \mathbf{P}_{1,a}^L \oplus \mathbf{P}_{2,a}^L, \mathbf{P}_{1,a}^U \oplus \mathbf{P}_{2,a}^U, r_{1,a}^L + r_{2,a}^L, r_{1,a}^U + r_{2,a}^U, L_1^\sharp \times L_2^\sharp \right)$$

where $(r_{1,a}^B + r_{2,a}^B)(s_1, s_2) = r_{1,a}^B(s_1) + r_{2,a}^B(s_2)$ for $B \in \{L, U\}$.

We can now present the main theorem of this section — that the ACTMC we obtain by composing the ACTMC components of a PEPA model is a safe approximation of the ACTMC obtained directly from the CTMC of the model. This is illustrated in Figure 3.

Theorem 4. Consider two PEPA components C_1 and C_2 , with abstractions (S_1^\sharp, α_1) and (S_2^\sharp, α_2) respectively. Let $\mathcal{M}_{i,a}^{\sharp\sharp} = \text{AbsComp}_{(S_i^\sharp, \alpha_i)}(\mathbf{Q}_a(C_i))$ for $i \in \{1, 2\}$. Then for all λ such that $\text{Unif}_\lambda(C_1 \boxtimes_L C_2)$ is defined, the following holds:

$$\text{Abs}_{(S^\sharp, \alpha)} \left(\text{Unif}_\lambda \left(\mathbf{Q} \left(C_1 \boxtimes_L C_2 \right) \right) \right) \leq \text{ACTMC}_\lambda \left(\sum_{a \in L} \mathcal{M}_{1,a}^{\sharp\sharp} \otimes \mathcal{M}_{2,a}^{\sharp\sharp} + \sum_{a \in \bar{L}} \mathcal{M}_{1,a}^{\sharp\sharp} \odot \mathcal{M}_{2,a}^{\sharp\sharp} \right)$$

where $S^\sharp = S_1^\sharp \times S_2^\sharp$, $\alpha(s_1, s_2) = (\alpha_1(s_1), \alpha_2(s_2))$ and $\bar{L} = (\text{Act}(C_1) \cup \text{Act}(C_2)) \setminus L$.

Since this method produces an over-approximation to the non-compositionally derived ACTMC, we can directly apply the model checking algorithm described in [3, 12] — this allows us to check transient three-valued CSL properties. In particular, given a CSL state property Φ , we can determine the set of states that definitely satisfy Φ (the model checker returns tt), and those that definitely do not (it returns ff).

Let us return once more to the PEPA model from Figure 2, and construct an ACTMC for the case when we aggregate states C_2 and C_3 . The Kronecker form of the generator matrix \mathbf{Q} of the model is as follows (the same as Equation 6, but without the term that evaluates to zero). To make the example concrete, we set the rates such that $r_a = r_2 = r_D = 1$ and $r_b = r_3 = 2$:

$$\begin{aligned} \mathbf{Q} = & \min \left\{ \begin{bmatrix} 0 \\ 2 \\ 4 \end{bmatrix}, \begin{bmatrix} \top \\ \top \end{bmatrix} \right\} \left(\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ & + \min \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ & + \min \left\{ \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{aligned}$$

The compositional ACTMC now has the following form, where we save space by writing intervals for the elements of the matrices, rather than intervals on the matrices:

$$\begin{aligned} \mathbf{Q}^\sharp = & \min \left\{ \begin{bmatrix} 0 \\ [2, 4] \end{bmatrix}, \begin{bmatrix} \top \\ \top \end{bmatrix} \right\} \left(\begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ & + \min \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ & + \min \left\{ \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \left(\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \end{aligned}$$

We can multiply this out to arrive at the following ACTMC (where the uniformisation constant $\lambda = 4$). For clarity, we have labelled the state that each row of the matrix

corresponds to:

$$Q^\# = \begin{matrix} (C_1 D_1) \\ (C_1 D_2) \\ (C_{\{2,3\}} D_1) \\ (C_{\{2,3\}} D_2) \end{matrix} \cdot 4 \left(\begin{bmatrix} \frac{3}{4} & 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ [\frac{1}{4}, \frac{1}{2}] & 0 & [\frac{1}{2}, \frac{3}{4}] & 0 \\ 0 & [\frac{1}{4}, \frac{1}{2}] & 0 & [\frac{1}{2}, \frac{3}{4}] \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

We are now in a position to model check transient CSL/X properties of this ACTMC, and compare them to the original PEPA model. As an example, consider the property $\mathcal{P}_{=?}(C_1 U^{[0,1]} C_{\{2,3\}})$, which asks the question “what is the probability that within the first time unit, we will remain in state C_1 before moving to state C_2 or C_3 ?” Since there are only two states in the abstracted C component, this is equivalent to asking whether we will leave state C_1 within the first time unit. Model checking the original model gives an answer of 0.6321, and in this case the ACTMC gives a precise answer of $[0.6321, 0.6321]$.

In this case, the abstraction is a success because it gives a very tight bound on the property. Of course, in general we cannot expect to always obtain tight bounds, and the choice of abstraction has a large impact on the precision. The purpose of this small example was to demonstrate how our abstraction is applied — we will look at a more interesting example in the next section, to illustrate this technique in practice.

5 An Example

To demonstrate the applicability of our technique, and how important the choice of abstraction is in obtaining precise bounds, we will consider a small example: The CEO of a large company often enjoys walking around the grounds of their corporate headquarters, whilst thinking up new marketing strategies. Yet as she strolls around, she still expects to receive emails from her secretary, courtesy of modern technology. A number of wireless access points are located in the grounds, but the signal strength varies from place to place. If an email with an urgent report must be downloaded as 9 separate packets, what is the probability that the CEO can expect to wait a certain length of time before receiving it?

$$\begin{aligned} L_{i,j} &= \sum_{(i',j') \in C(i,j)} (move, \top).L_{i',j'} + (download, r_D(i,j)).L_{i,j} \\ CEO_W &= (move, r_{walk}).CEO_W + (\tau, r_{stop}).CEO_T \\ CEO_T &= (\tau, r_{think}).CEO_T + (\tau, r_{start}).CEO_W \\ Device_i &= (download, r_{Dmax}).Device_{(i+1) \bmod 10} \end{aligned} \quad r_D = \begin{bmatrix} 10 & 5 & 2 & 1 & 1 \\ 5 & 5 & 2 & 2 & 1 \\ 2 & 2 & 5 & 5 & 5 \\ 1 & 2 & 5 & 10 & 5 \\ 1 & 1 & 5 & 5 & 5 \end{bmatrix}$$

$$CEO_W \xrightarrow[\{move\}]{} L_{0,0} \xrightarrow[\{download\}]{} Device_0$$

Fig. 4. A PEPA model of a wireless network

Figure 4 shows a PEPA model of such a scenario. We consider a 5×5 grid of locations $L_{i,j}$, such that $0 \leq i, j < 5$ and we define $C(i, j)$ to be the set of locations (i', j')

Aggregated States	State Space Size	$\mathcal{P}_{=?}(\text{tt } U^{[0,1]} \text{ Finished})$	$\mathcal{P}_{=?}(\text{tt } U^{[0,3]} \text{ Finished})$
None	500	[0.07847, 0.07847]	[0.74781, 0.74781]
$i, j \geq 4$	340	[0.07774, 0.08136]	[0.61874, 0.86546]
$i, j \geq 3$	200	[0.07556, 0.10893]	[0.49380, 0.96653]
Rows $L_{i,*}$	100	[0.00001, 0.18848]	[0.00001, 0.99185]
Columns $L_{*,j}$	100	[0.00001, 0.18848]	[0.00001, 0.99185]
Corners, Edges, Middle	60	[0.00001, 0.18848]	[0.02230, 0.99692]
All i, j	20	[0.00000, 0.27091]	[0.00380, 0.99890]

Table 1. Analysis of the PEPA model in Figure 4

adjacent to (i, j) — as an example, $C(0, 3) = \{(1, 3), (0, 2), (0, 4)\}$. $r_D(i, j)$ gives the rate of download at each location, and a particular configuration is shown to the right of the figure — with wireless access points at locations $(0, 0)$ and $(3, 3)$. The CEO is modelled by a component with two states — CEO_W corresponds to her walking, and CEO_T corresponds to her stopping and thinking. The mobile device cycles through states $Device_i$ for $0 \leq i < 10$, recording how many packets have been downloaded.

Table 1 shows some analysis results for this model², where the atomic proposition *Finished* corresponds to the device being in state $Device_9$. We analyse two properties, corresponding to the probability of completing the download within 1 minute and 3 minutes respectively. Given different abstractions of the $L_{i,j}$ component, it is clear that some result in much tighter bounds than others. Even the coarsest abstraction yields *some* useful information, however the tightest bounds in this case are when we aggregate the locations furthest from $L_{0,0}$ — we are less likely to reach these states within the specified time interval.

Although the state space reductions are relatively small in this example, remember that we are abstracting just one component in a small model. The major impact of our abstraction, in terms of state space reduction, is for models with many components in parallel, where we can abstract multiple components.

6 Conclusions

Abstract Markov chains are a powerful technique for reducing the size of a Markov chain, and allow us to obtain bounds on transient properties such as probabilistic reachability. We have applied this compositionally to PEPA models, allowing us to bound models where the underlying state space is too large to represent. We proved that the compositional abstraction yields a safe over-approximation of the non-compositional abstraction, and we demonstrated our technique with a small example.

Recently, abstract Markov chains have also been applied compositionally to Interval Markov Chains (IMC) in [11], through the use of modal transitions. In addition, there are many other techniques for abstracting Markov chains that we did not mention — for example, disaggregation/aggregation [15], quasi-lumpability [4], and stochastic bounds [6]. The main challenge, however, is in finding techniques that can be applied to as broad a class of model as possible.

² We take $r_{walk} = r_{stop} = r_{start} = 10$, $r_{think} = 1$, and $r_{Dmax} = 7$.

Whilst we have demonstrated the utility of compositional abstraction of PEPA models, there remain many interesting future research directions. One direction of particular interest is in algorithms for long-run averages over abstract Markov chains [1], which would allow the same abstractions to be used for CSL steady state formulae as for path formulae. In summary, abstract Markov chains are a useful technique for stochastic process algebra modellers to have at their disposal, and by bringing this to PEPA, we feel that we have broadened its practical applicability.

Acknowledgements We would like to thank Jane Hillston for her invaluable advice, and Joost-Pieter Katoen for his comments on an earlier version of this work.

References

1. L. De Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford, 1998.
2. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification (CAV)*, number 1855 in LNCS, pages 358–372. Springer, 2000.
3. C. Baier, H. Hermanns, J.-P. Katoen, and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science*, 345(1):2–26, 2005.
4. T. Dayar and W.J. Stewart. Quasi-lumpability, lower-bounding coupling matrices, and nearly completely decomposable Markov chains. *SIAM Journal on Matrix Analysis and Applications*, 18(2):482–498, 1997.
5. H. Fecher, M. Leucker, and V. Wolf. Don't know in probabilistic systems. In *Proceedings of SPIN'06*, number 3925 in LNCS, pages 71–88, 2006.
6. J.-M. Fourneau, M. Lecoq, and F. Quessette. Algorithms for an irreducible and lumpable strong stochastic bound. *Linear Algebra and its Applications*, 386:167–185, 2004.
7. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
8. J. Hillston and L. Kloul. An efficient Kronecker representation for PEPA models. In *Proceedings of the Joint International Workshop, PAPM-PROBMIV '01*, number 2165 in LNCS, pages 120–135. Springer, 2001.
9. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proceedings of TACAS 2006*, volume 3920 of LNCS, pages 441–444. Springer, 2006.
10. B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *LICS '91: Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 266–277, Amsterdam, The Netherlands, 1991.
11. J.-P. Katoen, D. Klink, and M.R. Neuhäuser. Compositional abstraction for stochastic systems. In *FORMATS '09: Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems*, pages 195–211. Springer-Verlag, 2009.
12. J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *Proceedings of 19th International Conference on Computer-Aided Verification (CAV'07)*, number 4590 in LNCS, pages 316–329. Springer, 2007.
13. J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer, 1976.
14. M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.
15. H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29(2):111–138, 1961.
16. M. Tribastone, A. Duguid, and S. Gilmore. The PEPA Eclipse plugin. *SIGMETRICS Performance Evaluation Review*, 36(4):28–33, 2009.

Theorem 1

Consider two generator matrices $\mathbf{Q}_1 = (r_1, \mathbf{P}_1)$ and $\mathbf{Q}_2 = (r_2, \mathbf{P}_2)$, corresponding to the same state space S (\mathbf{Q}_1 and \mathbf{Q}_2 are both $|S| \times |S|$ matrices). Then $\mathbf{Q}_1 + \mathbf{Q}_2$ can be written as follows:

$$\mathbf{Q}_1 + \mathbf{Q}_2 = (r_1, \mathbf{P}_1) + (r_2, \mathbf{P}_2) = \left(r_1 + r_2, \frac{r_1}{r_1 + r_2} \mathbf{P}_1 + \frac{r_2}{r_1 + r_2} \mathbf{P}_2 \right)$$

where $(r_1 + r_2)(s) = r_1(s) + r_2(s)$, 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 $s \in S$.

Proof. The proof is as follows. For an entry s, s' , when $s \neq s'$, in $\mathbf{Q}_1 + \mathbf{Q}_2$, we have:

$$\begin{aligned} (\mathbf{Q}_1 + \mathbf{Q}_2)(s, s') &= ((r_1, \mathbf{P}_1) + (r_2, \mathbf{P}_2))(s, s') \\ &= (r_1, \mathbf{P}_1)(s, s') + (r_2, \mathbf{P}_2)(s, s') \\ &= r_1(s) \mathbf{P}_1(s, s') + r_2(s) \mathbf{P}_2(s, s') \\ &= (r_1(s) + r_2(s)) \left(\frac{r_1(s)}{r_1(s) + r_2(s)} \mathbf{P}_1(s, s') + \frac{r_2(s)}{r_1(s) + r_2(s)} \mathbf{P}_2(s, s') \right) \\ &= \left(r_1 + r_2, \frac{r_1}{r_1 + r_2} \mathbf{P}_1 + \frac{r_2}{r_1 + r_2} \mathbf{P}_2 \right) (s, s') \end{aligned}$$

When $s = s'$, we similarly have:

$$\begin{aligned} (\mathbf{Q}_1 + \mathbf{Q}_2)(s, s) &= ((r_1, \mathbf{P}_1) + (r_2, \mathbf{P}_2))(s, s) \\ &= (r_1, \mathbf{P}_1)(s, s) + (r_2, \mathbf{P}_2)(s, s) \\ &= r_1(s)(\mathbf{P}_1(s, s) - 1) + r_2(s)(\mathbf{P}_2(s, s) - 1) \\ &= (r_1(s) + r_2(s)) \left(\frac{r_1(s)}{r_1(s) + r_2(s)} \mathbf{P}_1(s, s) + \frac{r_2(s)}{r_1(s) + r_2(s)} \mathbf{P}_2(s, s) - 1 \right) \\ &= \left(r_1 + r_2, \frac{r_1}{r_1 + r_2} \mathbf{P}_1 + \frac{r_2}{r_1 + r_2} \mathbf{P}_2 \right) (s, s) \end{aligned}$$

Theorem 2

For all well-formed PEPA models C , the CTMC induced by the semantics of PEPA and the CTMC described by the generator matrix $\mathbf{Q}(C)$, projected onto the derivative set $\text{ds}(C)$ (the reachable state space of C), are isomorphic.

Proof. Since, in a PEPA model, the transitions of each action type are independent from one another, we can consider them separately. We therefore need to prove that for each action type, the transition rates induced by the Kronecker form are identical to those induced by the PEPA semantics, as given in [7].

We proceed by induction on the structure of the system equation. In the base case, for a sequential component C_i , $\mathbf{Q}_a(C_i) = (r_{i,a}, \mathbf{P}_{i,a})$ corresponds, by Definition 9, precisely to those activities of type a that C_i can perform. In other words, for $s, s' \in S_i$, $r_{i,a}(s)$ is the apparent rate of action type a in state s , and $\mathbf{P}_{i,a}(s, s')$ is the relative probability of moving to state s' , if we perform an a activity in state s .

For the inductive case, we make the hypothesis that there is a transition $C_1 \xrightarrow{(a,r)} C_2$ induced by the PEPA semantics of a component C — where C may be a composition of sequential components, and $C_1, C_2 \in \text{ds}(C)$ — if and only if $\mathbf{Q}_a(C)(C_1, C_2) =$

$r_a(C_1)\mathbf{P}_a(C_1, C_2) = r$. We can ignore the case when $C_1 = C_2$ as self loops cancel out in the generator matrix.

Assume that there is an additional component C' such that $C'_1 \xrightarrow{(a, r')} C'_2$ iff it is the case that $\mathbf{Q}_a(C')(C'_1, C'_2) = r'$, as above. We will prove that for all sets of action types L , $C \bowtie_L C'$ induces a transition $C_1 \bowtie_L C'_1 \xrightarrow{(a, R)} C_2 \bowtie_L C'_2$ iff:

$$\mathbf{Q}_a(C \bowtie_L C')(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) = R$$

Consider the case $a \in L$. Then:

$$\begin{aligned} & \mathbf{Q}_a(C \bowtie_L C')(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= (\mathbf{Q}_a(C) \otimes \mathbf{Q}_a(C'))(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= ((r_a, \mathbf{P}_a) \otimes (r'_a, \mathbf{P}'_a))(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= (\min\{r_a, r'_a\}, \mathbf{P}_a \otimes \mathbf{P}'_a)(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= \min\{r_a(C_1), r'_a(C'_1)\}(\mathbf{P}_a(C_1, C_2) \times \mathbf{P}'_a(C'_1, C'_2)) \\ &= \min\{r_a(C_1), r'_a(C'_1)\} \frac{r}{r_a(C_1)} \frac{r'}{r'_a(C'_1)} \end{aligned}$$

where the final step follows from the induction hypothesis. This is equal by definition to the PEPA semantics of cooperation for action types $a \in L$, hence gives the rate R of the transition $C_1 \bowtie_L C'_1 \xrightarrow{(a, R)} C_2 \bowtie_L C'_2$ induced by the PEPA semantics.

Consider the case $a \notin L$. Then:

$$\begin{aligned} & \mathbf{Q}_a(C \bowtie_L C')(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= (\mathbf{Q}_a(C) \odot \mathbf{Q}_a(C'))(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= ((r_a, \mathbf{P}_a) \odot (r'_a, \mathbf{P}'_a))(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= ((r_a, \mathbf{P}_a) \otimes (r_\top, \mathbf{I}) + (r_\top, \mathbf{I}) \otimes (r'_a, \mathbf{P}'_a))(C_1 \bowtie_L C'_1, C_2 \bowtie_L C'_2) \\ &= \min\{r_a, \top\}(C_1, C'_1)\mathbf{P}_a(C_1, C_2)\mathbf{I}(C'_1, C'_2) + \min\{\top, r'_a\}(C_1, C'_1)\mathbf{I}(C_1, C_2)\mathbf{P}'_a(C'_1, C'_2) \\ &= r_a(C_1)\mathbf{P}_a(C_1, C_2)\mathbf{I}(C'_1, C'_2) + r'_a(C'_1)\mathbf{I}(C_1, C_2)\mathbf{P}'_a(C'_1, C'_2) \\ &= \begin{cases} r_a(C_1)\mathbf{P}_a(C_1, C_2) & \text{if } C'_2 = C'_1 \\ r'_a(C'_1)\mathbf{P}'_a(C'_1, C'_2) & \text{if } C_2 = C_1 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} r & \text{if } C'_2 = C'_1 \\ r' & \text{if } C_2 = C_1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where the final step follows from the induction hypothesis. This corresponds to the PEPA semantics of cooperation for action types $a \notin L$, where the activities of the two components take place independently. In other words, $C_1 \bowtie_L C'_1 \xrightarrow{(a, r)} C_2 \bowtie_L C'_1$ if $C_1 \xrightarrow{(a, r)} C_2$ in component C , and $C_1 \bowtie_L C'_1 \xrightarrow{(a, r')} C_1 \bowtie_L C'_2$ if $C'_1 \xrightarrow{(a, r')} C'_2$ in C' .

Theorem 3

Consider a CTMC $\mathcal{M} = (S, \mathbf{P}, r, L)$. For any uniformisation constant $\lambda \geq \max_{s \in S} r(s)$, and any abstraction (S^\sharp, α) on \mathcal{M} , the following holds:

$$\text{Abs}_{(S^\sharp, \alpha)}(\text{Unif}_\lambda(\mathcal{M})) \leq \text{ACTMC}_\lambda(\text{AbsComp}_{(S^\sharp, \alpha)}(\mathcal{M}))$$

Proof. Consider a CTMC $\mathcal{M} = (S, \mathbf{P}, r, L)$, which is not necessarily uniform. Let us define its uniformisation with respect to λ as:

$$\overline{\mathcal{M}} = \text{Unif}_\lambda(\mathcal{M}) = (S, \overline{\mathbf{P}}, \bar{r}, L)$$

where $\overline{\mathbf{P}}$ is the uniformised probability transition matrix (see Definition 3). Since $\overline{\mathcal{M}}$ is a uniformised CTMC, we can define its abstraction \mathcal{M}^\sharp with respect to (S^\sharp, α) as follows:

$$\mathcal{M}^\sharp = \text{Abs}_{(S^\sharp, \alpha)}(\overline{\mathcal{M}}) = (S^\sharp, \mathbf{P}^L, \mathbf{P}^U, \lambda, L^\sharp)$$

where \mathbf{P}^L and \mathbf{P}^U give the lower and upper bounds for the uniformised transition probabilities, and L^\sharp is the abstract labelling function (see Definition 8).

Considering the ACTMC component, let us define:

$$\mathcal{M}^{\sharp\sharp} = \text{AbsComp}_{(S^\sharp, \alpha)}(\mathcal{M}) = (S^\sharp, \mathbf{P}_C^L, \mathbf{P}_C^U, r_C^L, r_C^U, L^\sharp)$$

where \mathbf{P}_C^L , \mathbf{P}_C^U , r_C^L and r_C^U give the lower and upper bounds for the transition probabilities and exit rates, and L^\sharp is the abstract labelling function (see Definition 12). The CTMC induced by $\mathcal{M}^{\sharp\sharp}$ is given by:

$$\text{ACTMC}_\lambda(\mathcal{M}^{\sharp\sharp}) = (S^\sharp, \mathbf{P}'^L, \mathbf{P}'^U, \lambda, L^\sharp)$$

where \mathbf{P}'^L and \mathbf{P}'^U give upper and lower bounds for the uniformised transition probabilities, after converting the abstract CTMC component into an ACTMC (see Definition 11).

Let us now consider the lower bounding matrices — we require that for all $s, s' \in S^\sharp$, $\mathbf{P}'^L(s, s') \leq \mathbf{P}^L(s, s')$. Consider first the case when $s \neq s'$. Then we have:

$$\begin{aligned} \mathbf{P}'^L(s, s') &= \frac{r_C^L(s)}{\lambda} \mathbf{P}_C^L(s, s') && \text{Definition 11} \\ &= \frac{1}{\lambda} \min_{t \in \gamma(s)} r(t) \min_{t' \in \gamma(s')} \sum_{t'' \in \gamma(s')} \mathbf{P}(t, t'') && \text{Definition 12} \\ &\leq \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s')} \frac{r(t)}{\lambda} \mathbf{P}(t, t') && \text{Since } \min(ab) \geq \min(a) \min(b) \\ &= \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s')} \overline{\mathbf{P}}(t, t') && \text{Definition 3} \\ &= \mathbf{P}^L(s, s') && \text{Definition 8} \end{aligned}$$

Note that the central step works on the basis that all rates and probabilities are positive, hence the minimum of the product is greater than or equal to the product of the minima.

For the case when $s = s'$, we have:

$$\begin{aligned}
\mathbf{P}^{L}(s, s) &= 1 - \frac{r_C^U(s)}{\lambda} + \frac{r_C^L(s)}{\lambda} \mathbf{P}_C^L(s, s) && \text{Definition 11} \\
&= 1 - \frac{1}{\lambda} \max_{t \in \gamma(s)} r(t) + \frac{1}{\lambda} \min_{t \in \gamma(s)} r(t) \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s)} \mathbf{P}(t, t') && \text{Definition 12} \\
&\leq 1 - \frac{1}{\lambda} \max_{t \in \gamma(s)} r(t) + \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s)} \frac{r(t)}{\lambda} \mathbf{P}(t, t') \\
&\leq 1 - \max_{t \in \gamma(s)} \sum_{t' \in S \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t') + \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s) \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t') && \text{See Below} \\
&= \min_{t \in \gamma(s)} \left(\left(1 - \sum_{t' \in S \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t') \right) + \sum_{t' \in \gamma(s) \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t') \right) \\
&= \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s)} \bar{\mathbf{P}}(t, t') && \text{Definition 3} \\
&= \mathbf{P}^L(s, s) && \text{Definition 8}
\end{aligned}$$

To prove the noted step, let us assume that we have the minimum and maximum values, $t_1^{\min}, t_2^{\min}, t_3^{\max}$ and t_4^{\min} , of the following sums:

$$\begin{aligned}
- t_1^{\max} &\text{ maximises: } \max_{t \in \gamma(s)} (r(t)). \\
- t_2^{\min} &\text{ minimises: } \min_{t \in \gamma(s)} \sum_{t' \in \gamma(s)} \frac{r(t)}{\lambda} \mathbf{P}(t, t'). \\
- t_3^{\max} &\text{ maximises: } \max_{t \in \gamma(s)} \sum_{t' \in S \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t'). \\
- t_4^{\min} &\text{ minimises: } \max_{t \in \gamma(s)} \sum_{t' \in \gamma(s) \setminus \{t\}} \frac{r(t)}{\lambda} \mathbf{P}(t, t').
\end{aligned}$$

Using these minimising and maximising values, we have:

$$\begin{aligned}
&1 - \frac{r(t_1^{\max})}{\lambda} && + \sum_{t' \in \gamma(s)} \frac{r(t_2^{\min})}{\lambda} \mathbf{P}(t_2^{\min}, t') \\
\leq &1 - \frac{r(t_1^{\max})}{\lambda} && + \sum_{t' \in \gamma(s)} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t') \\
= &1 - \frac{r(t_1^{\max})}{\lambda} + \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t_4^{\min}) && + \sum_{t' \in \gamma(s) \setminus \{t_4^{\min}\}} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t') \\
\leq &1 - \frac{r(t_3^{\max})}{\lambda} + \frac{r(t_3^{\max})}{\lambda} \mathbf{P}(t_3^{\max}, t_3^{\max}) && + \sum_{t' \in \gamma(s) \setminus \{t_4^{\min}\}} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t') \\
= &1 - \frac{r(t_3^{\max})}{\lambda} (1 - \mathbf{P}(t_3^{\max}, t_3^{\max})) && + \sum_{t' \in \gamma(s) \setminus \{t_4^{\min}\}} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t') \\
= &1 - \frac{r(t_3^{\max})}{\lambda} \sum_{t' \in S \setminus \{t_3^{\max}\}} \mathbf{P}(t_3^{\max}, t') && + \sum_{t' \in \gamma(s) \setminus \{t_4^{\min}\}} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t') \\
= &1 - \sum_{t' \in S \setminus \{t_3^{\max}\}} \frac{r(t_3^{\max})}{\lambda} \mathbf{P}(t_3^{\max}, t') && + \sum_{t' \in \gamma(s) \setminus \{t_4^{\min}\}} \frac{r(t_4^{\min})}{\lambda} \mathbf{P}(t_4^{\min}, t')
\end{aligned}$$

Hence it holds that $P'^L(s, s') \leq P^L(s, s')$.

By a similar argument, we can show that $P^U(s, s') \leq P'^U(s, s')$, hence we have $\mathcal{M}^\sharp \leq ACTMC_\lambda(\mathcal{M}^{\sharp\sharp})$.

Theorem 4

Consider two PEPA components C_1 and C_2 , with abstractions (S_1^\sharp, α_1) and (S_2^\sharp, α_2) respectively. Let $\mathcal{M}_{i,a}^{\sharp\sharp} = AbsComp_{(S_i^\sharp, \alpha_i)}(\mathbf{Q}_a(C_i))$ for $i \in \{1, 2\}$. Then for all λ such that $Unif_\lambda(C_1 \boxtimes_L C_2)$ is defined, the following holds:

$$Abs_{(S^\sharp, \alpha)} \left(Unif_\lambda \left(\mathbf{Q} \left(C_1 \boxtimes_L C_2 \right) \right) \right) \leq ACTMC_\lambda \left(\sum_{a \in L} \mathcal{M}_{1,a}^{\sharp\sharp} \otimes \mathcal{M}_{2,a}^{\sharp\sharp} + \sum_{a \in \bar{L}} \mathcal{M}_{1,a}^{\sharp\sharp} \odot \mathcal{M}_{2,a}^{\sharp\sharp} \right)$$

where $S^\sharp = S_1^\sharp \times S_2^\sharp$, $\alpha(s_1, s_2) = (\alpha_1(s_1), \alpha_2(s_2))$, and $\bar{L} = (Act(C_1) \cup Act(C_2)) \setminus L$.

Proof. Consider first a particular action type $a \in L \cup \bar{L}$. This results in the following term from the above comparison (expanding out the Kronecker operator on the left hand side):

$$Abs_{(S^\sharp, \alpha)} (Unif_\lambda (\mathbf{Q}_a(C_1) \oplus \mathbf{Q}_a(C_2))) \leq ACTMC_\lambda (\mathcal{M}_{1,a}^{\sharp\sharp} \oplus \mathcal{M}_{2,a}^{\sharp\sharp})$$

Where $\oplus = \otimes$ if $a \in L$, and \odot if $a \in \bar{L}$.

From Theorem 3, we have the following:

$$Abs_{(S^\sharp, \alpha)} (Unif_\lambda (\mathbf{Q}_a(C_1) \oplus \mathbf{Q}_a(C_2))) \leq ACTMC_\lambda (AbsComp_{(S^\sharp, \alpha)} (\mathbf{Q}_a(C_1) \oplus \mathbf{Q}_a(C_2)))$$

We therefore need to show that:

$$AbsComp_{(S^\sharp, \alpha)} (\mathbf{Q}_a(C_1) \oplus \mathbf{Q}_a(C_2)) = \mathcal{M}_{1,a}^{\sharp\sharp} \oplus \mathcal{M}_{2,a}^{\sharp\sharp}$$

Consider the case when $a \in L$, and therefore $\oplus = \otimes$. We have:

$$\mathbf{Q}_a(C_1) \otimes \mathbf{Q}_a(C_2) = (S_1 \times S_2, \mathbf{P}_{1,a} \otimes \mathbf{P}_{2,a}, \min\{r_{1,a}, r_{2,a}\}, L_1 \times L_2)$$

The ACTMC component $\mathcal{M}_a^{\sharp\sharp}$ that this induces is as follows:

$$\begin{aligned} \mathcal{M}_a^{\sharp\sharp} &= AbsComp_{(S^\sharp, \alpha)} (\mathbf{Q}_a(C_1) \otimes \mathbf{Q}_a(C_2)) \\ &= (S_1^\sharp \times S_2^\sharp, (\mathbf{P}_{1,a} \otimes \mathbf{P}_{2,a})^L, (\mathbf{P}_{1,a} \otimes \mathbf{P}_{2,a})^U, (\min\{r_{1,a}, r_{2,a}\})^L, (\min\{r_{1,a}, r_{2,a}\})^U, L_1^\sharp \times L_2^\sharp) \end{aligned}$$

But notice that the lower bound $(\mathbf{P}_{1,a} \otimes \mathbf{P}_{2,a})^L$ is the same as $\mathbf{P}_{1,a}^L \otimes \mathbf{P}_{2,a}^L$, since the minimum of a product is the same as the product of the minima, for positive values. Furthermore, the lower bound for the rate function, $(\min\{r_{1,a}, r_{2,a}\})^L$, is the same as $\min\{r_{1,a}^L, r_{2,a}^L\}$, the minimum of the lower bounding rate functions. The same holds

for the upper bounds. But this is the same as the composition of the ACTMC components of C_1 and C_2 for action type a :

$$\mathcal{M}_{1,a}^{\#\#} \otimes \mathcal{M}_{2,a}^{\#\#} = \left(S_1^{\#} \times S_2^{\#}, \mathbf{P}_{1,a}^L \otimes \mathbf{P}_{2,a}^L, \mathbf{P}_{1,a}^U \otimes \mathbf{P}_{2,a}^U, \min\{r_{1,a}^L, r_{2,a}^L\}, \min\{r_{1,a}^U, r_{2,a}^U\}, L_1^{\#} \times L_2^{\#} \right)$$

It therefore follows that $\mathcal{M}_a^{\#\#} = \mathcal{M}_{1,a}^{\#\#} \otimes \mathcal{M}_{2,a}^{\#\#}$.

Consider the case when $a \in \bar{L}$, and therefore $\oplus = \odot$. We have:

$$\mathbf{Q}_a(C_1) \odot \mathbf{Q}_a(C_2) = (S_1 \times S_2, \mathbf{P}_{1,a} \oplus \mathbf{P}_{2,a}, r_{1,a} + r_{2,a}, L_1 \times L_2)$$

This induces the following ACTMC component:

$$\begin{aligned} \mathcal{M}_a^{\#\#} &= \text{AbsComp}_{(S^{\#}, \alpha)}(\mathbf{Q}_a(C_1) \odot \mathbf{Q}_a(C_2)) \\ &= \left(S_1^{\#} \times S_2^{\#}, (\mathbf{P}_{1,a} \oplus \mathbf{P}_{2,a})^L, (\mathbf{P}_{1,a} \oplus \mathbf{P}_{2,a})^U, (r_{1,a} + r_{2,a})^L, (r_{1,a} + r_{2,a})^U, L_1^{\#} \times L_2^{\#} \right) \end{aligned}$$

But we have, for:

$$\begin{aligned} (r_{1,a} + r_{2,a})^L(s_1^{\#}, s_2^{\#}) &= \min_{s_1 \in \gamma(s_1^{\#}), s_2 \in \gamma(s_2^{\#})} r_{1,a}(s_1) + r_{2,a}(s_2) \\ &= \min_{s_1 \in \gamma(s_1^{\#})} r_{1,a}(s_1) + \min_{s_2 \in \gamma(s_2^{\#})} r_{2,a}(s_2) \\ &= r_{1,a}^L(s_1^{\#}) + r_{2,a}^L(s_2^{\#}) \end{aligned}$$

The same follows for the upper bound of the rate function, and we follow a similar argument for the bounds of the probabilistic transition matrices. Hence this is the same as the composition of the ACTMC components of C_1 and C_2 for action type a :

$$\mathcal{M}_{1,a}^{\#\#} \oplus \mathcal{M}_{2,a}^{\#\#} = \left(S_1^{\#} \times S_2^{\#}, \mathbf{P}_{1,a}^L \oplus \mathbf{P}_{2,a}^L, \mathbf{P}_{1,a}^U \oplus \mathbf{P}_{2,a}^U, r_{1,a}^L + r_{2,a}^L, r_{1,a}^U + r_{2,a}^U, L_1^{\#} \times L_2^{\#} \right)$$

Hence $\mathcal{M}_a^{\#\#} = \mathcal{M}_{1,a}^{\#\#} \oplus \mathcal{M}_{2,a}^{\#\#}$.

We have shown that the safety of the abstraction is preserved for all action types $a \in L \cup \bar{L}$, and so it follows that this also holds for the sum over all action types.