An Aggregation Technique For Large-Scale PEPA Models With Non-Uniform Populations

Alireza Pourranjbar , Jane Hillston
LFCS, School of Informatics, University of Edinburgh
a.pourranjbar@sms.ed.ac.uk , jane.hillston@ed.ac.uk

ABSTRACT

Performance analysis based on modelling consists of two major steps: model construction and model analysis. Formal modelling techniques significantly aid model construction but can exacerbate model analysis. In particular, here we consider the analysis of *large-scale* systems which consist of one or more entities replicated many times to form large populations. The replication of entities in such models can cause their state spaces to grow exponentially to the extent that their exact stochastic analysis becomes computationally expensive or even infeasible.

In this paper, we propose a new *approximate aggregation* algorithm for a class of large-scale PEPA models. For a given model, the method quickly checks if it satisfies a *syntactic* condition, indicating that the model may be solved approximately with high accuracy. If so, an aggregated CTMC is generated directly from the model description. This CTMC can be used for efficient derivation of an approximate marginal probability distribution over some of the model's populations. In the context of a large-scale client-server system, we demonstrate the usefulness of our method.

Categories and Subject Descriptors

C.4 [**Performance of systems**]: Modeling techniques

General Terms

Theory, Performance

1. INTRODUCTION

Discrete state modelling is a very expressive style of modelling which has been used in a number of domains, including performance modelling, where it is supported by a number of modelling formalisms such as queueing networks, stochastic Petri nets and stochastic process algebras. These formalisms ease the task of model construction but can exacerbate the underlying disadvantage of discrete state modelling, the problem of state space explosion. In this paper we present a new aggregation technique for a class of Markovian-based models expressed in the stochastic process algebra PEPA [7]. This formalism supports compositionality: first the behaviours of individual components are defined and then these are composed to form the system's complete description. The aggregation approach we develop takes advantage of this feature.

In this paper, we consider a class of *large-scale* PEPA models, i.e. models in which there exist one or more components which are instantiated many times to form large populations. In such models, the interactions between individual components can be seen as interactions between the corresponding populations. In particular we focus on models in which component populations have significantly different sizes — some components appear in large populations, but other components belong to populations which are relatively small. The models in this class reflect many resource-bound computer and communication networks, which typically consist of two types of entities, *resources* and *resource users*. In such systems it is often the case that a rather large population of the users are served by a smaller population of the resources.

Having constructed the model of a large-scale system, theoretically a wide variety of analysis techniques can be applied for their performance evaluation. At one extreme, numerical analysis would construct the associated infinitesimal generator matrix and obtain the model's probability distribution across its complete state space. Whilst it provides a full and faithful account of the system behaviour this route often proves to be infeasible due to the size of the state space. At the other end of the spectrum, analysis based on fluid approximation or mean field techniques can be used to derive the probability distribution's first few moments rather than the full distribution [6]. These moments can provide the basis for performance estimation and the fluid flow analysis has specifically been shown to be useful for models with uniformly large populations [12]. However, applying the method to our sub-class of models can give rise to misleading results [10], because the fluid flow moments can be too abstract or crude to reflect the system's full distribution (e.g. the distribution may be heavy-tailed or multi-modal).

Here we present an approach which is particularly tailored for studying resource-bound systems in which large populations of users interact with a limited population of resources. Our method is based on a state space aggregation and the information it provides about the complete probability dis-

ValueTools'13, December 10 – 12 2013, Turin, Italy

VALUETOOLS 2013, December 10-12, Torino, Italy

DOI 10.4108/icst.valuetools.2013.254397

tribution for resource populations is more detailed and fine-grained than the fluid flow moments. At the same time, it avoids the construction of the model's potentially very large CTMC. We show that for a large-scale model in this sub-class, the aggregation is possible if the model satisfies a syntactic condition which can be readily checked, and from the aggregated CTMC certain performance measures can be readily derived.

Paper's structure: in Section 2, we introduce the syntax and state space representation used when dealing with large-scale PEPA models. In Section 3 we formally introduce the sub-class and illustrate an aggregation condition. In Sections 4 and 5 we show the aggregation steps and the way a marginal probability distributions is derived. In Section 6, we describe the usefulness of the method in the context of an example. Section 7 presents the related work. In Section 8 we conclude and describe plans for extending the method.

2. PRELIMINARIES

In this section we give a brief introduction to the PEPA modelling language. The interested reader is referred to [7, 6] for more detail. In PEPA, models are constructed of components which undertake activities. Each activity (α, r) has a specified activity type, α, and a rate, r, which is assumed to govern an exponentially distributed delay determining the duration of the activity. Sequential components are composed concurrently, but may be constrained to share particular activities. In some cases the rate of an activity may be unspecified, or *passive*, denoted $\top$, indicating that a component is willing to participate in a shared activity with no constraint on the rate. Here, we use the *grouped* PEPA syntax for clear specification of large-scale PEPA models [6].

$$S = (\alpha, r).S \mid S + S \mid C_s \qquad P = P \underset{L}{\bowtie} P \mid S$$
$$D = D \mid\mid D \mid P \qquad M = M \underset{L}{\bowtie} M \mid Y\{D\}$$

Here S represents a *sequential* component (essentially, a state machine). These components are built up using *prefix* (.): designated first activity, and *choice* (+): preemptive alternative behaviours. C_s is a constant used to name a component. P, a *model* component, is composed of sequential components $(S \underset{L}{\bowtie} S)$ which are constrained to share activities whose type is in the set L. When L is empty these components may proceed independently, denoted by $P \mid\mid Q$. Syntactic sugar $P[n]$ is used to denote the parallel composition of n identical P processes, $P[n] = \underbrace{P \mid\mid P \mid\mid \ldots \mid\mid P}_{n \text{ times}}$. D represents the *parallel* composition of instances of a sequential component. These instances constitute a group and a unique label is assigned to each group, denoted by Y. M represents the composition of groups. We restrict ourselves to grouped PEPA models in which all groups are strictly *simple*; i.e. each group contains the instances of one sequential component only. Here is an example of such a model.

Model 1. This model captures the dynamics of a client-server system.

$$C_{think} \stackrel{def}{=} (think, r_t).C_{req} \qquad C_{req} \stackrel{def}{=} (req, r_c).C_{think}$$
$$S_{idle} \stackrel{def}{=} (req, r_s).S_{log} + (brk, r_b).S_{broken}$$
$$S_{log} \stackrel{def}{=} (log, r_l).S_{idle} \qquad S_{broken} \stackrel{def}{=} (fix, r_f).S_{idle}$$
$$CS \stackrel{def}{=} Servers\ \{\ S_{idle}[2]\ \} \underset{\{req\}}{\bowtie} Clients\ \{\ C_{think}[2]\ \}$$

Description: C_{think} and C_{req} are the local states of the Client sequential component: clients alternate between thinking and requesting. Similarly S_{idle}, S_{log} and S_{broken} define the behaviour of the Server sequential component. In state S_{idle} a server offers the shared activity *req*, followed by logging in state S_{log}. Alternatively the server might suffer a breakdown (*brk*) and enter the state S_{broken} until it is fixed.

The last line of the model is called the *system equation* and corresponds to M in the grammar. This specifies how the sequential components are instantiated in groups and the cooperation pattern between groups. Here, the model is initialised with two instances of idle servers and two instances of thinking clients. The servers form group *Servers* and the group of the clients is labelled *Clients*. The cooperation set specifies that instances in group *Servers* synchronise on *req* activities with instances in group *Clients*. Moreover, there is no synchronisation between the instances of servers or clients within their respective groups.

The grouped PEPA structured operational semantics can be used to build the models' underlying transition relations [6]. The relations can help when generating the model's underlying stochastic process, readily seen to be a CTMC.

For a grouped PEPA model $\mathbb{M}$, let $\mathcal{G}(\mathbb{M})$ denote its set of group labels. This can be calculated recursively:

$$\mathcal{G}(\mathbb{M}) = \begin{cases} \mathcal{G}(M_1) \cup \mathcal{G}(M_2) & \text{if } \mathbb{M} = M_1 \underset{L}{\bowtie} M_2 \\ Y & \text{if } \mathbb{M} = Y\{D\} \end{cases} \quad (1)$$

Let $N^h_{\mathbb{M}}$ denote the number of groups in $\mathbb{M}$. We define $\mathcal{C}_{\mathbb{M}}$ to be the set of sequential components defined in the model. The number of distinct sequential components in the model is denoted by $N^c_{\mathbb{M}}$. Given that all groups in $\mathbb{M}$ are simple, we can construct the function $sc : \mathcal{G}(\mathbb{M}) \to \mathcal{C}_{\mathbb{M}}$ which relates a group's label to the sequential component whose instances form that group[1]. For example, $sc(Servers) = S_{idle}$.

For a $C \in \mathcal{C}_{\mathbb{M}}$, let $ds(C)$ denote the set of local states or local derivatives C visits. We assume that the number of local states the sequential component C_x experiences is denoted by N_{C_x}. As an example, $ds(S_{idle})$=$\{S_{idle}, S_{log}, S_{broken}\}$. Accordingly, we define $ds^*(H)$, the set of local states each instance in the group labelled H experiences. For $H \in \mathcal{G}(\mathbb{M})$: $ds^*(H)$=$ds(sc(H))$. N_{H_i} denoted the cardinality of $ds^*(H)$.

In the system equation of a model, the operator $\underset{L}{\bowtie}$ is used to compose the model's groups and form cooperations. The instances of two groups which are composed are restricted to synchronize on action types in the associated cooperation set. These groups, however, can be subjected to further compositions and synchronizations with other groups of the model. Thus, in a model, the system equation introduces a *hierarchy of cooperation* among its different groups and assigns to each group, the set of action types the instances in that group must synchronize on. To clarify, consider the process $(G_1\{\cdot\} \underset{L'}{\bowtie} G_2\{\cdot\}) \underset{L}{\bowtie} (G_3\{\cdot\} \underset{L''}{\bowtie} G_4\{\cdot\})$. Here G_1 and G_2 cooperate on set L'; similarly, G_3 and G_4 cooperate on actions in set L''. Additionally, these component groups are

[1]We assume that a sequential component is represented by its initial state.

also combined in cooperation on set L. The cooperation on L' restricts the instances in groups G_1 and G_2 and does not affect G_3 or G_4; but all instances are restricted by L.

The notion of cooperation hierarchy in a model $\mathbb{M}$ can be formally captured by a partial order relation $<^*_{\mathbb{M}}$. For group compositions M_i and M_j, $M_i <^*_{\mathbb{M}} M_j$, if and only if M_i is composed when constructing M_j. If $M_i <^*_{\mathbb{M}} M_j$, all cooperation sets applied to M_j are also enforced on M_i. The following rules construct $<^*_{\mathbb{M}}$ using $\mathbb{M}$'s system equation.

1. $H\{\cdot\} <^*_{\mathbb{M}} H\{\cdot\}$ 2. $X <^*_{\mathbb{M}} A$, if $A \stackrel{def}{=} X$
3. $H\{\cdot\} <^*_{\mathbb{M}} M_i\{\cdot\} \underset{L}{\bowtie} M_j\{\cdot\}$, if $H <^*_{\mathbb{M}} M_i \vee H <^*_{\mathbb{M}} M_j$

For a group $H \in \mathcal{G}(\mathbb{M})$, $\mathcal{I}(\mathbb{M}, H)$ denotes the set of action types on which instances in H are required to synchronize. $\mathcal{I}(\mathbb{M}, H)$ is defined recursively from the system equation, using the subsidiary function $\mathcal{J}$; $\mathcal{I}(\mathbb{M}, H) = \mathcal{J}(\mathbb{M}, H, \emptyset)$.

$$\mathcal{J}(M, H, K) = \begin{cases} K, & \text{if } M = H\{\cdot\} \\ \mathcal{J}(M_1, H, K \cup L) \cup \mathcal{J}(M_2, H, K \cup L), & \\ & \text{if } M = M_1 \underset{L}{\bowtie} M_2. \\ \mathcal{J}(A, H, K), & \text{if } M \stackrel{def}{=} A. \end{cases}$$

We also define $sync(\mathbb{M}, H, \alpha)$, the set of groups in $\mathbb{M}$ whose instances synchronize on α activities with instances in H. Using $<^*_{\mathbb{M}}$, $sync(\mathbb{M}, H, \alpha)$ can be found as follows:

$$sync(M,H,\alpha) = \begin{cases} \emptyset, & \text{if } M = H\{\cdot\} \\ sync(M_1, H, \alpha) \cup \{H_i \in \mathcal{G}(\mathbb{M}) \mid H_i\{\cdot\} <^*_{\mathbb{M}} M_2\}, & \\ \quad \text{if } M = M_1 \underset{L}{\bowtie} M_2, \alpha \in L, H <^*_{\mathbb{M}} M_1 & \\ sync(M_2, H, \alpha) \cup \{H_i \in \mathcal{G}(\mathbb{M}) \mid H_i\{\cdot\} <^*_{\mathbb{M}} M_1\}, & \\ \quad \text{if } M = M_1 \underset{L}{\bowtie} M_2, \alpha \in L, H <^*_{M} M_2 & \\ sync(M_1, H, \alpha), & \\ \quad \text{if } M = M_1 \underset{L}{\bowtie} M_2, \alpha \notin L, H <^*_{\mathbb{M}} M_1 & \\ sync(M_2, H, \alpha), & \\ \quad \text{if } M = M_1 \underset{L}{\bowtie} M_2, \alpha \notin L, H <^*_{\mathbb{M}} M_2 & \end{cases}$$

State Space Representation. In the context of large-scale PEPA models, we are often content to abstract from the behaviour of individuals and focus on the evolution of populations [12]. Thus, in order to capture the current state of a group H, we do not capture the state of each individual instance but instead, for each local state $C_{x,y}$ in $ds^*(H)$, with a counter $\xi(H, C_{x,y})$ we record how many instances in H exhibit the behaviour associated with $C_{x,y}$ and form the vector $\boldsymbol{\xi}(H) = \langle \xi(H, C_{x,y}) \mid C_{x,y} \in ds^*(H) \rangle$ to capture the distribution of instances in H across the local states $ds^*(H)$. Having formed $\boldsymbol{\xi}(H)$ for each group, we construct $\boldsymbol{\xi} = \langle \boldsymbol{\xi}(H) \mid H \in \mathcal{G}(\mathbb{M}) \rangle$, the vector which captures the state of all groups in the model. The model's *dimension*, $d_{\mathbb{M}} = \sum_{H \in \mathcal{G}(\mathbb{M})} N_H$, is defined to be the number of state variables appearing in its state vector.

Each counter $\xi(H, C_{x,y})$ can be regarded as a random variable and therefore, the state vector is regarded as a vector of random variables. The CTMC capturing the model's evolution is based on the numerical state vector. At any given time, a joint probability distribution can be associated with this CTMC.

Let C_u be a local state of a sequential process in $\mathcal{C}_{\mathbb{M}}$. The *apparent rate* of an action type α in C_u, denoted by $r_\alpha(C_u)$, is the total rate C_u offers for α activities [7]. Note that C_u might have the form: $C_u = (\alpha, r').C'_u + (\alpha, r'').C''_u + \ldots$ with multiple states reachable from C_u via an α activity.

$$r_\alpha(C_u) = \begin{cases} r, & \text{if } C_u = (\beta, r).P \ , \ \beta = \alpha \\ 0, & \text{if } C_u = (\beta, r).P \ , \ \beta \neq \alpha \\ r_\alpha(P) + r_\alpha(Q), & \text{if } C_u = P + Q \end{cases}$$

We define $r_\alpha(C_u, C'_u)$ to be the apparent rate of an action type α in C_u leading specifically to derivative C'_u.

$$r_\alpha(C_u, C'_u) = \begin{cases} r, & \text{if } C_u = (\beta, r).P \wedge \beta = \alpha \wedge P = C'_u \\ 0, & \text{if } C_u = (\beta, r).P \wedge (\beta \neq \alpha \vee P \neq C'_u) \\ r_\alpha(P, C'_u) + r_\alpha(Q, C'_u), & \text{if } C_u = P + Q \end{cases}$$

Note that C_u might be passive with respect to α. Moreover, C_u might enable more than one passive α activity. In such a case, each is given a weight ($\omega \in \mathbb{N}$), clarifying the relative probability assigned to these activities. These are handled appropriately in the apparent rate arithmetic, but we omit details here due to space limitations.

Let us assume that in $H_i \in \mathcal{G}(\mathbb{M})$ some instances are in state $C_u \in ds^*(H_i)$, enabling α activities, where $\alpha \notin \mathcal{I}(\mathbb{M}, H_i)$. Undertaking the α activity by any of these instances causes $\xi(H_i, C_u)$ to decrease by one and the number of instances in one of the one-step α derivatives to increase by one.

$$\langle \boldsymbol{\xi}(H_1), \ldots, \boldsymbol{\xi}(H_i), \ldots, \boldsymbol{\xi}(H_{N^h_{\mathbb{M}}}) \rangle \xrightarrow{(\alpha, R_\alpha(\boldsymbol{\xi}(H_i)))} \langle \boldsymbol{\xi}(H_1), \ldots, \boldsymbol{\xi}'(H_i), \ldots, \boldsymbol{\xi}(H_{N^h_{\mathbb{M}}}) \rangle$$

Assume that the state change $C_u \xrightarrow{(\alpha, r_\alpha)} C'_u$ occurs. Then:

$$\boldsymbol{\xi}(H_i) = \langle \ldots, \xi(H_i, C_u), .., \xi(H_i, C'_u), .., \xi(H_i, C_{N_{H_i}}) \rangle$$

$$\boldsymbol{\xi}'(H_i) = \langle \ldots, \xi(H_i, C_u) - 1, .., \xi(H_i, C'_u) + 1, .., \xi(H_i, C_{N_{H_i}}) \rangle$$

For $H \in \mathcal{G}(\mathbb{M}), C_u, C'_u \in ds^*(H)$, $r^*_\alpha(H, C_u, C'_u)$ denotes the apparent rate of action type α in the context of H restricted to activities which move from state C_u to C'_u.

$$r^*_\alpha(H, C_{x,y}, C'_{x,y}) = \xi(H, C_{x,y}) \times r_\alpha(C_{x,y}, C'_{x,y})$$

According to the PEPA semantics, the rate of the transition above is: $R_\alpha(\boldsymbol{\xi}(H_i)) = \xi(H, C_{x,y}) \times r^*_\alpha(H_i, C_u, C'_u)$. We also define $r^*_\alpha(H)$, the apparent rate of α in H offered by all instances in H; $r^*_\alpha(H) = \sum_{C \in ds(sc(H))} \xi(H, C) \times r_\alpha(C)$. When instances in two distinct groups cooperate (say, C_u instances in H_i cooperate with C_v instances in H_j on a shared α activity becoming C'_u and C'_v respectively) the update to the numerical state vector will make changes analogous to those outlined above, i.e. the subvectors corresponding to the two groups. Then, the rate of the shared activity will be

$$R_\alpha(\boldsymbol{\xi}(H_i), \boldsymbol{\xi}(H_j)) = \frac{r^*_\alpha(H_i, C_u, C'_u)}{r^*_\alpha(H_i)} \frac{r^*_\alpha(H_j, C_v, C'_v)}{r^*_\alpha(H_j)} \min(r^*_\alpha(H_i), r^*_\alpha(H_j)). \quad (2)$$

3. APPROXIMATE AGGREGATION

3.1 Syntactic Condition

We consider a sub-class of large-scale PEPA models in which the populations of different sequential components have significantly different sizes. Consider a model $\mathbb{M}$ in this sub-class and assume that according to their sizes, $\mathbb{M}$'s groups are partitioned into two sets, large groups and small groups. Let $\mathcal{G}_s(\mathbb{M})$ denote the set of small groups in $\mathbb{M}$ and $\mathcal{G}_l(\mathbb{M})$ denote its set of large groups: $\mathcal{G}(\mathbb{M}) = \mathcal{G}_s(\mathbb{M}) \cup \mathcal{G}_l(\mathbb{M})$.

This set partition will also partition $\mathbb{M}$'s state vector — the state variables related to the groups in $\mathcal{G}_s(\mathbb{M})$ and those related to the groups in $\mathcal{G}_l(\mathbb{M})$. Without loss of generality, the model's state vector can be written as $\langle \boldsymbol{\xi}_s, \boldsymbol{\xi}_l \rangle$ where $\boldsymbol{\xi}_s$ corresponds to the model's small groups, and $\boldsymbol{\xi}_l$ to the large groups.

As $\mathbb{M}$ is assumed to be a large-scale model, its complete CTMC is very large and its construction and analysis are computationally expensive. We show that if $\mathbb{M}$ satisfies the following condition, then we can perform an *approximate* aggregation which results in an aggregated CTMC for $\mathbb{M}$. We illustrate how this step enables us to efficiently estimate a marginal probability distribution over the model's small groups with a high accuracy. The cost of finding such a distribution using the aggregated CTMC is orders of magnitude smaller than through the analysis of the original CTMC.

Condition 1. Syntactic Aggregation Condition For any shared activity, synchronised between one or more large groups and one or more small group, the rate of the shared activity, should be completely decided by the small groups:

$$\forall\, H^l \in \mathcal{G}_l(\mathbb{M})\ \ \forall\, \alpha \in \mathcal{I}_{\mathcal{A}}(M,H)$$
$$[\ (\ sync(M,H^l,\alpha)\ \cap\ \mathcal{G}_s(\mathbb{M})\ ) \neq \emptyset\ \Longrightarrow$$
$$\forall\, C_{x,y}\ \in\ ds^*(H^l)\ ,\ \exists\, \omega \in \mathbb{N}\ ,\ \ r_\alpha(C_{x,y}) = \omega\top\ ]$$

The condition expresses that in any synchronization on a shared activity of type α, if both small and large groups are involved, then all instances in the involved large groups need to undertake α passively. Consider a variant of the client-server model with the client components modified as:

$$C_{think} \stackrel{def}{=} (think, r_t).C_{req} \qquad C_{req} \stackrel{def}{=} (req, \top).C_{think}$$

We assume *Clients* constitutes a large group, and *Servers*, a small one. They synchronise on the *req* activity which the clients undertake passively. Thus the model now satisfies Condition 1. We will see that this implies that in any state of the system, the global apparent rate of *req* depends only on the configuration of the servers and is independent of the clients' configuration.

3.2 The CTMC Structure

The CTMC of a large-scale model with small groups that satisfies Condition 1 exhibits important structural properties which can be exploited in order to build an aggregated CTMC. We introduce these properties and illustrate them in the CTMC of the modified client-server model initialised with two servers and two clients.

Given that the model's groups are partitioned by size, we can categorise each transition depending on whether it changes the state of only one or more small groups, the state of only one or more large groups or simultaneously, the state of both small and large groups. For $H \in \mathcal{G}(\mathbb{M})$, let us define $\vec{\mathcal{A}}^*(H)$ to be the set of action types enabled by H and $\vec{\mathcal{A}}^*_i(H) = \vec{\mathcal{A}}^*(H) - \mathcal{I}(\mathbb{M},H)$ represents the set of action types of individual activities undertaken by instances in H. First, assume that $H \in \mathcal{G}_l(\mathbb{M})$ and define $\vec{\mathcal{A}}^*_l(H)$ to be the set of action types instances in H undertaken individually or in cooperation with other large groups.

$$\vec{\mathcal{A}}^*_l(H) = \{\ \alpha \mid \alpha \in \vec{\mathcal{A}}^*_i(H) \vee (sync(M,H,\alpha) \cap \mathcal{G}_s(\mathbb{M})) = \emptyset\ \}$$

Based on $\vec{\mathcal{A}}^*_l(H)$, we define $\vec{\mathcal{A}}^*_l(\mathbb{M}) = \bigcup_{H \in \mathcal{G}_l(\mathbb{M})} \vec{\mathcal{A}}^*_l(H)$, the set of action types related to the dynamics of the large groups only. As the second case, assume that $H \in \mathcal{G}_s(\mathbb{M})$ and define $\vec{\mathcal{A}}^*_s(H)$ analogously for small groups:

$$\vec{\mathcal{A}}^*_s(H) = \{\ \alpha \mid \alpha \in \vec{\mathcal{A}}^*_i(H) \vee (sync(M,H,\alpha) \cap \mathcal{G}_l(\mathbb{M})) = \emptyset\ \}$$

As above, we define $\vec{\mathcal{A}}^*_s(\mathbb{M}) = \bigcup_{H \in \mathcal{G}_s(\mathbb{M})} \vec{\mathcal{A}}^*_s(H)$ which represents the action types in which the large groups do not participate. Finally, for $H \in \mathcal{G}_s(\mathbb{M})$, we define $\vec{\mathcal{A}}^*_{sl}(H)$ to represent the set of action types shared between instances of H and instances of one or more large groups.

$$\vec{\mathcal{A}}^*_{sl}(H) = \{\ \alpha \mid \alpha \in \mathcal{I}(\mathbb{M},H) \wedge (sync(M,H,\alpha) \cap \mathcal{G}_l(\mathbb{M})) \neq \emptyset\}$$

Similarly, for model $\mathbb{M}$, $\vec{\mathcal{A}}^*_{sl}(\mathbb{M}) = \bigcup_{H \in \mathcal{G}_s(\mathbb{M})} \vec{\mathcal{A}}^*_{sl}(H)$ is defined to be the set of action types in which both small and large groups participate. This categorisation of the action types will be used to characterise the structural properties exhibited by the CTMC of the models satisfying Condition 1. For the modified client-server system, $\vec{\mathcal{A}}^*_l(CS) = \{think\}$, $\vec{\mathcal{A}}^*_{sl}(CS) = \{req\}$ and $\vec{\mathcal{A}}^*_s(CS) = \{log, brk, fix\}$.

3.2.1 Identifiable Sub-Chains

Consider a model $\mathbb{M}$ satisfying Condition 1. In its underlying CTMC, let us focus on transitions of type $\vec{\mathcal{A}}^*_l(\mathbb{M})$. These transitions only change the state of one or more large groups, leaving the configuration of the small groups unchanged. Based on this, the CTMC can be divided into sub-chains where the states within each sub-chain are connected by $\vec{\mathcal{A}}^*_l(\mathbb{M})$ transitions and for which the configuration of instances in the small groups remains the same. From a state S, a sub-chain Y_i, can be derived using these rules:

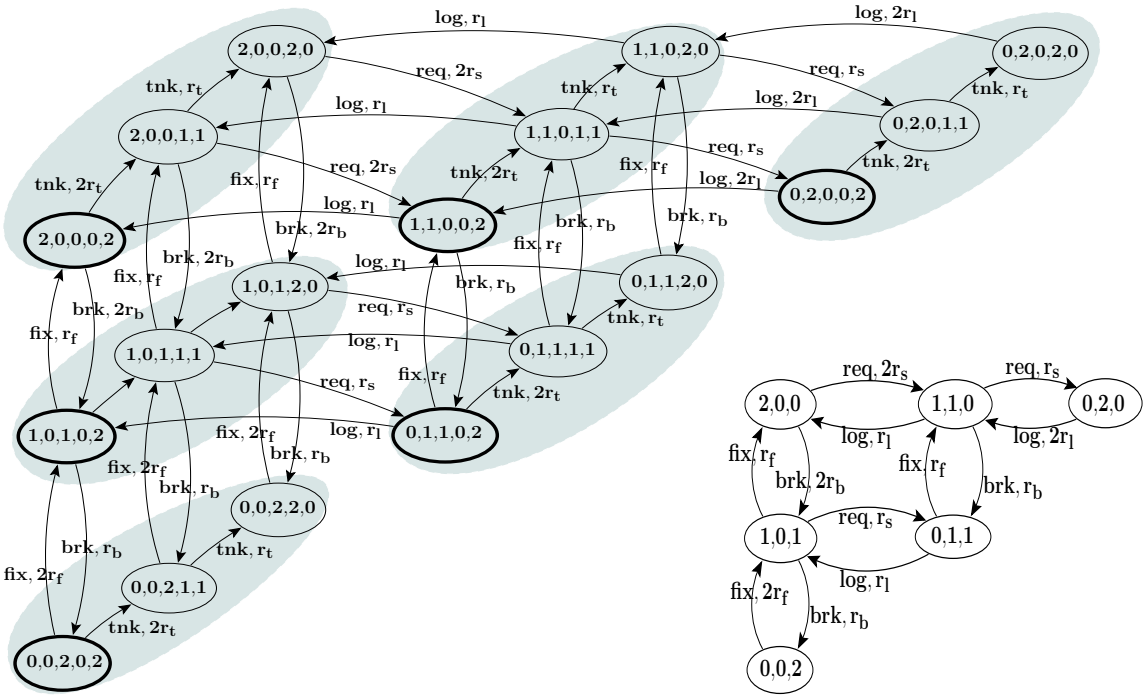
$$S \in Y_i\ \wedge\ S \xrightarrow{(\alpha,\cdot)} S'\ \wedge\ \alpha \in \vec{\mathcal{A}}^*_l(\mathbb{M})\ \Longrightarrow\ S' \in Y_i$$
$$S \in Y_i\ \wedge\ S'' \xrightarrow{(\alpha,\cdot)} S\ \wedge\ \alpha \in \vec{\mathcal{A}}^*_l(\mathbb{M})\ \Longrightarrow\ S'' \in Y_i$$

The sub-chain Y_i associated with a state S_i consists of S_i, the states from which S_i can be reached by $\vec{\mathcal{A}}^*_l(\mathbb{M})$ transitions, and the states reachable from S by such transitions. These transitions capture the dynamics of the large groups in Y_i.

Using the rules above, given $\mathcal{G}_s(\mathbb{M})$ and $\mathcal{G}_l(\mathbb{M})$, for $\mathbb{M}$'s underlying CTMC, the partition $Y_{\mathbb{M}} = \{Y_1, Y_2, \ldots\}$ can be formed where each Y_i is a sub-chain which can be uniquely identified by the configuration it captures for $\mathcal{G}_s(\mathbb{M})$ (the vector $\boldsymbol{\xi}_s$). Two states S_1 and S_2 in the original CTMC belong to two different sub-chains if they do not exhibit the same configuration for instances of groups in $\mathcal{G}_s(\mathbb{M})$. In Fig. 1A, the CTMC of the modified model, the sub-chains are visually identified. In each sub-chain, the clients change their state via *think* activities without affecting the servers. For a model with a larger client population, the same partition, but with longer sub-chains, would be observable.

3.2.2 Unlikely Boundary States, Blocked Activities

Before showing the next structural property, we need to define the notion of boundary states. Within the CTMC of $\mathbb{M}$, consider an arbitrary sub-chain Y_i and $\alpha \in \vec{\mathcal{A}}^*(M)$. First, let us assume that $\alpha \in \vec{\mathcal{A}}^*_l(M)$, i.e. small groups do not par-



(A) Complete CTMC. (B) Aggregated CTMC.

FIGURE 1: Complete and aggregated CTMC for the client-server system with two servers and two clients (shorthand $tnk = think$ is used). Each state of the complete CTMC is a vector $\langle S_i, S_l, S_b, C_r, C_t\rangle$, where S_i counts the number of idle servers, S_l the number of logging servers, S_b the number of broken servers, C_r the number of requesting clients and C_t the number of thinking clients. Each state of the aggregated CTMC follows this state descriptor: $\langle S_i, S_l, S_b\rangle$. Boundary states are highlighted.

ticipate in α activities. In this case, in any state $S \in Y_i$, whether S enables α activities or not, depends only on the configuration of the large groups in S. As the second case, assume that $\alpha \in \vec{\mathcal{A}}^*_s(\mathbb{M})$. Here, the large groups do not participate in the α activities and therefore, in any state $S \in Y_i$, whether α is enabled depends only on the configuration S captures for the small groups. Moreover, since the configuration of the small groups is identical across all the states in Y_i, the status of α is the same across Y_i as a whole.

The third case is when $\alpha \in \vec{\mathcal{A}}^*_{sl}(\mathbb{M})$. Here, α activities are shared between one or more small and one or more large groups and the status of α in any state depends on the configurations of groups of both types. Here we can identify a subset of states in Y_i, where shared α activities are not enabled because there are no instances in one or more relevant large groups to participate. Extending this to all action types in $\vec{\mathcal{A}}^*_{sl}(\mathbb{M})$, in each sub-chain Y_i, we can find a subset of states where one or more action types α remain disabled specifically due to the lack of cooperation from the instances in the large groups. For Y_i, let us refer to this subset as the *boundary states* of Y_i, denoted $bo(Y_i)$. For instance, in the client-server example, the boundary states are those where there are no clients ready to make a request. The formal characterisation of the boundary states follows:

Definition 1. In $\mathbb{M}$'s state space, a state S with the vector $\boldsymbol{\xi} = \langle \boldsymbol{\xi}(H_1), \ldots, \boldsymbol{\xi}(H_{N^H_{\mathbb{M}}})\rangle$, is a boundary state if and only if:

$$\exists\ H_s \in \mathcal{G}_s(\mathbb{M})\ \ \exists\ \alpha \in (\vec{\mathcal{A}}^*(H_s) \cap \vec{\mathcal{A}}^*_{sl}(\mathbb{M}))$$
$$\exists\ H_l \in (\mathcal{G}_l(\mathbb{M}) \cap sync(M, H_s, \alpha))\ :\quad r^*_\alpha(H_l) = 0$$

We define $bl(Y_i, \alpha)$, $Y_i \in Y_{\mathbb{M}}, \alpha \in \vec{\mathcal{A}}^*_{sl}(\mathbb{M})$, to represent the set of states in Y_i where α is blocked.

$$bl(Y_i, \alpha) = \{S \mid S \in Y_i\ \wedge\ \alpha\ \in bl'(S)\}$$
$$bl'(S) = \Big\{\alpha \mid \exists H_s \in \mathcal{G}_s(\mathbb{M}) : \Big[\ \alpha \in (\vec{\mathcal{A}}^*(H_s) \cap \vec{\mathcal{A}}^*_{sl}(\mathbb{M})) \wedge$$
$$\exists\ H_l \in (\mathcal{G}_l(\mathbb{M})\ \cap sync(M, H_s, \alpha)) : r^*_\alpha(H_l) = 0\ \Big]\Big\}$$

In large-scale resource-bound systems, resources are unlikely to remain idle waiting for the resource users to ask for the service. Therefore, in such systems, the probability of being in the boundary states can be assumed to be negligible.

3.2.3 Rate Regularities

Consider $\mathbb{M}$, a model satisfying Condition 1. In the underlying CTMC, consider a sub-chain Y_i and state $S_i \in Y_i$. Assume that there exists a cross-sub-chain transition $S_i \xrightarrow{(\alpha,R)} S_j$ where $S_j \in Y_j, Y_j \neq Y_i$ (see Fig. 2A). Then for any α enabling state $S'_i \in Y_i$, we can observe a similar transition $S'_i \xrightarrow{(\alpha,R')} S'_j$ where $S'_j \in Y_j$ and $R = R'$. This observation leads to the following Lemma.

Lemma 3.1. Rate regularity for non-boundary states.

$$\forall\ \alpha \in (\vec{\mathcal{A}}^*_{sl}(\mathbb{M}) \cup\ \vec{\mathcal{A}}^*_s(\mathbb{M}))\ ,\quad \forall\ S_i\ \in\ Y_i\ ,\quad \forall\ S_j\ \in\ Y_j$$
$$\Big(\ S_i \xrightarrow{(\alpha,R)} S_j\ \Longrightarrow\ \forall\ S'_i\ \in\ (Y_i - bl(Y_i, \alpha))$$
$$\Big[\ r^*_\alpha(S'_i > 0)\ \Longrightarrow\ \exists\ S'_j\ \in\ Y_j\quad S'_i \xrightarrow{(\alpha,R)} S'_j\ \Big]\ \Big)$$

In any sub-chain, all non-boundary states enable the same set of activities. Hence, for $\alpha \in \vec{\mathcal{A}}^*_{sl}(\mathbb{M}) \cup \vec{\mathcal{A}}^*_{s}(\mathbb{M})$, we can define $r(\alpha, Y_i, Y_j)$, the rate of any α transition from any $S_i \in Y_i$ to another state $S_j \in Y_j$.

$$r(\alpha, Y_i, Y_j) = \begin{cases} r & \text{if } \exists\, S_i \in Y_i\ \exists\, S_j \in Y_j :\ S_i \xrightarrow{(\alpha,r)} S_j \\ 0 & \text{otherwise} \end{cases}$$

Between any two sub-chains Y_i and Y_j, transitions of more than one type might connect their states (see Fig. 2A). We define $r(Y_i, Y_j) = \sum_{\alpha \in (\vec{\mathcal{A}}^*_s(\mathbb{M}) \cup \vec{\mathcal{A}}^*_{sl}(\mathbb{M}))} r(\alpha, Y_i, Y_j)$ to represent the total rate at which non-boundary states in Y_i transition into a corresponding state in Y_j.

In Fig. 1A, the activities of type $\vec{\mathcal{A}}^*_s(CS) \cup \vec{\mathcal{A}}^*_{sl}(CS) = \{req, log, brk, fix\}$ cause the system to leave the sub-chain it resides in and move into a new one. The rate of these activities depends only on the configuration of the servers. For the states within one sub-chain, the state of the servers does not change. Thus, the rate of any activity of type $\vec{\mathcal{A}}^*_s(CS) \cup \vec{\mathcal{A}}^*_{sl}(CS)$, if enabled, is the same.

For a large-scale model $\mathbb{M}$ that satisfies Condition 1, we can construct its CTMC and use $\vec{\mathcal{A}}^*_s(\mathbb{M})$, $\vec{\mathcal{A}}^*_l(\mathbb{M})$, and $\vec{\mathcal{A}}^*_{sl}(\mathbb{M})$ to detect the sub-chains formed and observe the rate regularities for transitions between sub-chains. These regularities and the assumption that the probability of experiencing the boundary states is negligible, enable us to build an aggregated CTMC for the model. In this CTMC, each sub-chain is represented by a single aggregate state. This aggregate state abstracts away the dynamics of the large groups in that sub-chain and only captures the un-changed configuration the sub-chain exhibits for the small groups. In other words, the aggregated CTMC captures only the evolution of the small groups and information about the behaviour of the instances in the large groups is lost. Nevertheless, the aggregate CTMC provides the means to study the evolution of the instances of the groups in $\mathcal{G}_s(\mathbb{M})$ in a fined-grained manner without building the very large original CTMC. Fig. 1B shows the aggregated CTMC of the modified client-server model. It shows how the state variables related to the servers behave.

4. AGGREGATION ALGORITHM

The aggregated CTMC of a conforming model $\mathbb{M}$ is more abstract than its original CTMC (capturing only the state and transitions of the instances in $\mathcal{G}_s(\mathbb{M})$) and its structure is independent of the size of the large group populations. Therefore, we propose an algorithm which builds the aggregated CTMC *directly* from the model. At first, in a reduction step, our algorithm transforms the system equation of the original model $\mathbb{M}$ into $\mathbb{M}_R$, a *reduced* form which captures $\mathbb{M}$'s structure only with respect to the groups in $\mathcal{G}_s(\mathbb{M})$. In the next step, using $\mathbb{M}_R$ and a semantics developed for PEPA population models, the aggregated CTMC is generated. These steps are described in this section. Note that the reduced system equation faithfully captures the synchronization restrictions imposed on any of the small groups; the reduction rules guarantee the behaviour exhibited considering $\mathbb{M}_R$ matches the one seen for $\mathcal{G}_s(\mathbb{M})$ in the original CTMC.

4.1 Reduction

These rules will be applied to the system equation of an input model $\mathbb{M}$ in order to produce the reduced model $\mathbb{M}_R$.

$$red(G) = \begin{cases} red(G_1) \underset{L}{\bowtie} red(G_2), & \text{if } G = G_1 \underset{L}{\bowtie} G_2 \\ H\{\cdot\}, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_s(\mathbb{M}) \\ Nil, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_l(\mathbb{M}) \\ red(X), & \text{if } G \stackrel{def}{=} X \end{cases}$$

The process Nil represents a sequential process which does not undertake any activity. The following rules remove Nil processes to find the minimal reduced system equation:

$$Nil \underset{\cdot}{\bowtie} Nil = Nil \qquad Nil \underset{\cdot}{\bowtie} P = P \qquad P \underset{\cdot}{\bowtie} Nil = P$$

4.2 Count-Oriented Semantics

Having built the reduced form of $\mathbb{M}$'s system equation, we apply the *count-oriented structured operational semantics* presented below to derive the model's underlying labelled transition system (LTS) *directly* in numerical vector form.

Promotion. Promotion of a sequential component's transition to the group level:

$$\frac{C_u \xrightarrow{(\alpha, r_\alpha)} C'_u}{\boldsymbol{\xi}(H) \xrightarrow{(\alpha, \xi(H,C_u) r_\alpha(C_u, C'_u))} \Theta(\boldsymbol{\xi}(H), C_u, C'_u)}\ \xi(H, C_u) > 0$$

Cooperation. Cooperation between groups:

$$\frac{\boldsymbol{\xi}(H_i) \overset{(\alpha, r(\boldsymbol{\xi}(H_i)))}{\rightsquigarrow} \boldsymbol{\xi}'(H_i)}{\boldsymbol{\xi}(H_i) \underset{L}{\bowtie} \boldsymbol{\xi}(H_j) \overset{\alpha, r(\boldsymbol{\xi}(H_i)))}{\rightsquigarrow} \boldsymbol{\xi}'(H_i) \underset{L}{\bowtie} \boldsymbol{\xi}(H_j)}\ \alpha \notin L$$

$$\frac{\boldsymbol{\xi}(H_i) \overset{(\alpha, r_1(\boldsymbol{\xi}(H_i)))}{\rightsquigarrow} \boldsymbol{\xi}'(H_i)\ \wedge\ \boldsymbol{\xi}(H_j) \overset{(\alpha, r_2(\boldsymbol{\xi}(H_j)))}{\rightsquigarrow} \boldsymbol{\xi}'(H_j)}{\boldsymbol{\xi}(H_i) \underset{L}{\bowtie} \boldsymbol{\xi}(H_j) \overset{(\alpha, R)}{\rightsquigarrow} \boldsymbol{\xi}'(H_i) \underset{L}{\bowtie} \boldsymbol{\xi}'(H_j)}\ \alpha \in L$$

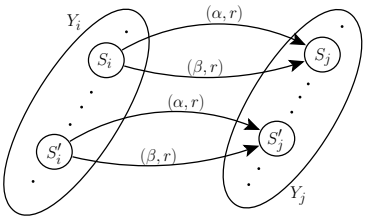
$$R = \frac{r_1(\boldsymbol{\xi}(H_i))}{r^*_\alpha(H_i)} \frac{r_2(\boldsymbol{\xi}(H_j))}{r^*_\alpha(H_j)} min(r^*_\alpha(H_i), r^*_\alpha(H_j))$$

Constant. Process constants:

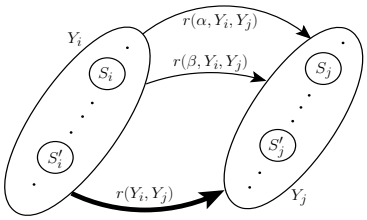
$$\frac{M \xrightarrow{(\alpha, r)} M'}{A \xrightarrow{(\alpha, r)} M'}\ A \stackrel{def}{=} M$$

Due to the space limitation, the symmetric rule for the first cooperation rule (evolution of the right hand process on individual activities) is omitted. The rules for constructing the transition relation $\rightsquigarrow$ make use of the transition relation $\rightarrow$ built by the original PEPA semantics [7] (see the premise of the first rule). However, note that $\rightarrow$ needs to be constructed only at the level of the sequential processes; for each sequential process in the model an automata is constructed which captures the states and transitions each instance of that sequential process experiences.

The semantics formally reflects the update to the numerical vector representation of the states, as described in Section 2, recording the impact of the completion of an action on each of the counts for the involved groups. The appro-



(A) The non-boundary states in a sub-chain presenting similar behaviours.



(B) The rate for any one cross-sub-chain transition.

FIGURE 2: Rate regularities for cross-sub-chain transitions.

priate update is determined by the function Θ.

$$\Theta(\boldsymbol{\xi}(H), C_u, C'_u) = \langle \Theta'(\boldsymbol{\xi}(H), C_u, C'_u, \xi(H, C''_u)) \,|\, C''_u \in ds^*(H) \rangle$$

$$\Theta'(\boldsymbol{\xi}(H), C_u, C'_u, \xi(H, C''_u)) = \begin{cases} \xi(H, C'') - 1, & \text{if } C'' = C_u \\ \xi(H, C'') + 1, & \text{if } C'' = C'_u \\ \xi(H, C''), & \text{otherwise} \end{cases}$$

4.3 Generating the Aggregated CTMC

In the aggregation algorithm the count-oriented semantics is applied to the reduced form system equation of a model, creating its aggregated LTS. For a model $\mathbb{M}$, this LTS, which is formally characterised below, constitutes the model's aggregated CTMC.

Aggregated Derivative Set. The aggregated derivative set of a state S, denoted by $D^*(S)$, is the smallest set of states satisfying the following conditions.

1. $S \in D^*(S)$.
2. If $S_1 \in D^*(S)$ and $S_1 \overset{(\alpha,\cdot)}{\rightsquigarrow} S_2$, then $S_2 \in D^*(S)$.

Aggregated Labelled Transition System is defined as a tuple $(D^*(S_0), \Omega, \rightsquigarrow)$. S_0 is the model's initial aggregate state. $\Omega = (\mathcal{A} \times \mathcal{F})$ is the transition system's alphabet where $\mathcal{A}$ is the set of action types defined in $\mathbb{M}$ and $\mathcal{F}$ is a function space defined over $\mathbb{Z}_+^{d_{\mathbb{M}_R}}$. $d_{\mathbb{M}_R}$ is the dimension of the reduced system equation, i.e. the number of state variables appearing in its underlying state vector. Each function in $\mathcal{F}$ corresponds to an action type: F_α defines the rate of observing an activity of type α in the vector space $\mathbb{Z}_+^{d_{\mathbb{M}_R}}$.

Fig. 1B shows the aggregated CTMC constructed for the modified Model 1 when initialised with two idle servers.

5. DERIVING MARGINAL DISTRIBUTIONS

Derivation through the aggregated CTMC. The numerical state vector of a PEPA model can be regarded as a vector of dependent random variables whose evolution is captured by the underlying CTMC. For this CTMC, the Chapman-Kolmogorov (C-K) equations can be derived for the evolution of the joint probability distribution of the state random variables over time. For any model, the number of C-K equations is equal to the size of the state space of the CTMC. Therefore, as explained earlier, calculating the full joint probability distributions for large-scale PEPA models is usually infeasible.

Recall that the state vector $\boldsymbol{\xi}$ of a large-scale PEPA model $\mathbb{M}$ satisfying the aggregation conditions can be expressed as $\boldsymbol{\xi} = \langle \boldsymbol{\xi}_s, \boldsymbol{\xi}_l \rangle$ where $\boldsymbol{\xi}_s$ captures the state of instances within groups in $\mathcal{G}_s(\mathbb{M})$ and $\boldsymbol{\xi}_l$, the state of the instances within groups in $\mathcal{G}_l(\mathbb{M})$. The aggregation algorithm constructs an aggregated CTMC in which only the evolution of $\boldsymbol{\xi}_s$ is captured. This CTMC can be used to derive *marginal* probability distributions over the random variables $\xi(H, C_{x,y}), \forall H_i \in \mathcal{G}_s(\mathbb{M}), \forall C_{x,y} \in ds^*(H_i)$. These are calculated by a set of ordinary differential equations (ODE). In the following, assume that X_a denotes the stochastic process whose behaviour is captured by the aggregated CTMC. For each state of this CTMC, one equation is constructed:

```
for each aggregate state S in ds*(S0) form one
equation d P(Xa=S)/dt as E

ins ← in(S) = {Sh | Sh ⇝(α,·) S }
outS ← out(S) = {Sj | S ⇝(α,·) Sj}
    for each state Sh ∈ ins : Sh ⇝(α,r*α(ξ)) S
       //probability flux into S
       add  + r*α(ξ) × P(Xa = Sh)  to E

    for each state Sj ∈ outS : S ⇝(α,r*α(ξ)) Sj
      //probability flux out of S
        add  − r*α(ξ) × P(Xa = S)  to E
```

PROGRAM 1: Pseudo-code that generates the ODEs for a model's marginal probability distributions.

$$\frac{d\,\mathbb{P}(X_a = S)}{d\,t} = \sum_{\alpha : S_j \overset{(\alpha, r^*_\alpha(\boldsymbol{\xi}))}{\rightsquigarrow} S} r^*_\alpha(\boldsymbol{\xi})\, \mathbb{P}(X_a = S_j) - \sum_{\alpha : S \overset{(\alpha, r^*_\alpha(S))}{\rightsquigarrow} S_j} r^*_\alpha(S)\, \mathbb{P}(X_a = S)$$

The algorithm in Program. 1 constructs this system of ODEs.

Derivation of the marginal distribution from the C-K equations. For a PEPA model, its system of C-K equations can be constructed by forming an equation for each state of the model, recording the flux in and the flux out of that state. This allows analysis at the level of individual states. In this section, we show a high level overview of how it is possible to construct the equations for evaluating the probability of being in each sub-chain using the model's original C-K equations.

Let X be the random process captured by $\mathbb{M}$'s original CTMC and $\mathbb{P}_{\langle X = S_i \rangle}(t)$ denote the probability that at time t, X is in state S_i. In the model's C-K equations, one equation

is constructed for each state X experiences:

$$\frac{\mathbb{P}_{\langle X=S_i\rangle}(t)}{d\ t} = \sum_{S_j \xrightarrow{(\alpha, r_{\langle\alpha,S_j,S_i\rangle})} S_i,\ S_j \in Y_{\mathbb{M}}} r_{\langle\alpha,S_j,S_i\rangle}\mathbb{P}_{\langle X=S_j\rangle}(t) - \sum_{S_i \xrightarrow{(\alpha, r_{\langle\alpha,S_i,S_j\rangle})} S_j\ ,\ S_j \in Y_{\mathbb{M}}} r_{\langle\alpha,S_i,S_j\rangle}\mathbb{P}_{\langle X=S_i\rangle}(t) \quad (3)$$

The probability of being in a sub-chain Y_i at time t is equal to the sum of the probabilities of being in any of the states in Y_i: $\mathbb{P}_{\langle X\subset Y_i\rangle}(t) = \sum_{S_i\in Y_i} \mathbb{P}_{\langle X=S_i\rangle}(t)$. To study these probability evolutions, we form one equation for each sub-chain Y_i. $\mathbb{P}_{\langle X\subset Y_i\rangle}(t)$ denotes the probability that at time t the system is in any of the states in Y_i.

$$\frac{d\ \mathbb{P}_{\langle X\subset Y_i\rangle}(t)}{d\ t} = \frac{d\ \sum_{S_i\in Y_i}\mathbb{P}_{\langle X=S_i\rangle}(t)}{d\ t} = \sum_{S_i\in Y_i}\frac{d\ \mathbb{P}_{\langle X=S_i\rangle}(t)}{d\ t} \quad (4)$$

In the context of Y_i, the probability fluxes which correspond to action types in $\overrightarrow{\mathcal{A}}_l^*(\mathbb{M})$ cancel each other and have no effect on the probability fluxes in and out of the whole sub-chain. Substituting (3) into (4), using Lemma 3.1 and taking advantage of the assumption that the probability of being in boundary states is close to zero, we derive, $\forall\ Y_i \in Y_{\mathbb{M}}$:

$$\frac{d\ \mathbb{P}_{\langle X\subset Y_i\rangle}(t)}{d\ t} \approx -\sum_{Y_j\in Y_{\mathbb{M}}} r(Y_i,Y_j)\mathbb{P}_{\langle X\subset Y_i\rangle}(t) + \sum_{Y_k\in Y_{\mathbb{M}}} r(Y_k,Y_i)\mathbb{P}_{\langle X\subset Y_k\rangle}(t)$$

which captures the probability evolutions at the level of sub-chains. The complete proof is described in [9].

For a large-scale PEPA model which satisfies Condition 1, we have shown two approaches to deriving equations leading to the corresponding marginal probability distributions over sub-chains; first, constructing the equations by using the aggregated CTMC and second, by converting the model's original C-K equations. The two routes lead to the same equations since each sub-chain is represented as a single state in the aggregate CTMC and this CTMC is faithful in capturing the probability flows into and out of the sub-chains. However, note that the cost of deriving the marginal distributions through the aggregated CTMC is orders of magnitude smaller. In the next section, we consider an experiment where the accuracy of the approximate aggregate ODEs is investigated.

6. ACCURACY OF THE AGGREGATION

For a model which satisfies Condition 1, the accuracy of the marginal probability distributions derived via the aggregation, depends on the assumption that the probability mass in the boundary states is close to zero. If, at all times, the small groups in the model are under heavy load and their cooperation capacity is saturated by the demand of the large groups, one would expect to get highly accurate approximate marginal probability distributions. Conversely, if the probability of being in boundary states is not negligible, then the approximation method may lead to erroneous marginal distributions. In this section, we report on experiments to investigate the accuracy of the aggregation method in the context of the client-server model.

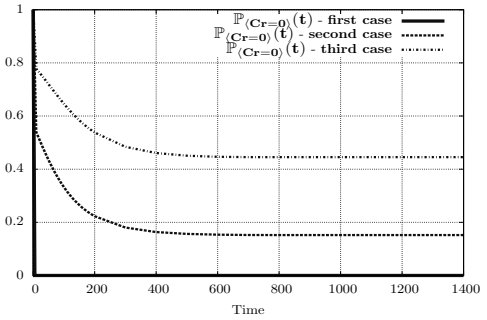


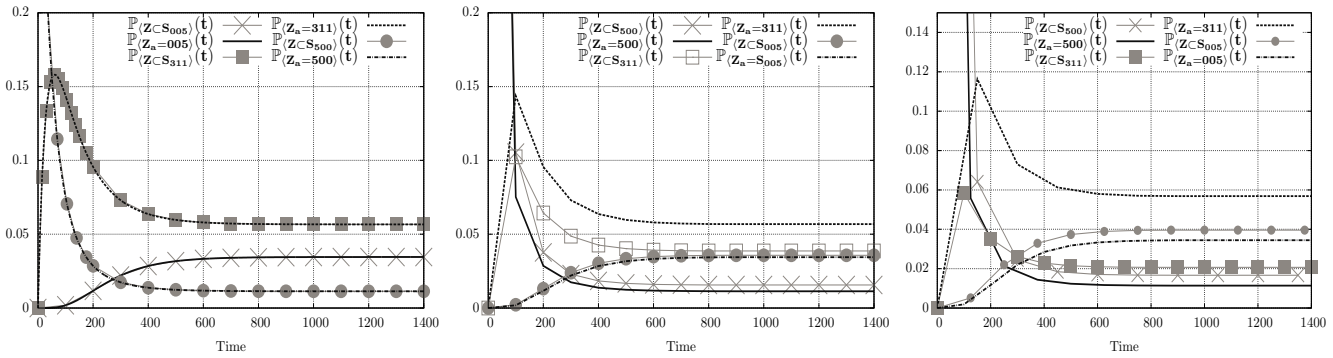
FIGURE 3: Probability of being in boundary states ($\mathbb{P}_{\langle C_r=0\rangle}(t)$) for each case of the experiment.

We consider Model 1 and assume that the small group, *Servers*, contains 5 servers and the large group, *Clients*, 100 clients. In the model's *original* CTMC, which captures the dynamics of both clients and servers, the boundary states are those where the number of clients requesting service is zero (i.e. $C_r = 0$). We constructed three versions of this model and across these, the following parameters were the same: $n_s = 5$, $n_c = 100$, $r_s = 10$, $r_l = 50$, $r_b = 0.005$, $r_f = 0.005$ and $r_c = \top$. However, in the first version $r_t = 15$, in the second $r_t = 0.2$ and in the third $r_t = 0.1$. This change in r_t causes a gradual increase in the probability of being in boundary states (see Fig. 3). For each version, we calculated an approximate marginal probability distribution over the servers and compared it with a similar distribution derived from the exact analysis of the model's complete CTMC using the PRISM tool [8]. There are multiple ways for comparing two probability distributions. For simplicity, we chose three different representative states from the distributions and compared the distributions only with respect to those particular states. Our comparison could readily be extended to the complete distributions. In this section, Z denotes the stochastic process representing the client-server model's original CTMC and Z_a denotes the stochastic process associated with the model's aggregated CTMC.

The parameters chosen for the first version ($r_t = 15$) cause the servers to be under heavy contention at all times; i.e. probability of $C_r = 0$ is close to zero. Figure (4A) shows a comparison between probabilities calculated for three representative states, $\langle 5,0,0\rangle$, $\langle 3,1,1\rangle$ and $\langle 0,0,5\rangle$, by the approximate and exact methods. As an example, $\mathbb{P}_{\langle Z\subset(3,1,1)\rangle}(t)$, denotes the probability that in the original CTMC, the system resides in a state where there are three idle, one logging and three broken servers and $\mathbb{P}_{\langle Z_a=(3,1,1)\rangle}(t)$, the probability that the aggregated CTMC resides in an equivalent state.

In the second case, $r_t = 0.2$, thinking has a longer duration which slows the flow of clients into the state of requesting communication.Thus, the probability of $C_r = 0$ becomes higher. The same measures were calculated for the second case and the results are reported in Fig. (4B). Here, the probability of being in the boundary states is higher resulting in a less accurate marginal distribution. In the third case $r_t = 0.1$, the probability of $C_r = 0$ is relatively high (see Fig. 3). Hence, the deviation of the approximate distribution from the exact one is significantly larger than the previous cases. The outputs for this case are shown in Fig (4C).

The aggregated CTMC can be used for deriving further



(A) $\mathbb{P}_{\langle Z\subset(5,0,0)\rangle}, \mathbb{P}_{\langle Z\subset(3,1,1)\rangle}, \mathbb{P}_{\langle Z\subset(0,0,5)\rangle}$ first case.
(B) $\mathbb{P}_{\langle Z\subset(5,0,0)\rangle}, \mathbb{P}_{\langle Z\subset(3,1,1)\rangle}, \mathbb{P}_{\langle Z\subset(0,0,5)\rangle}$ second case.
(C) $\mathbb{P}_{\langle Z\subset(5,0,0)\rangle}, \mathbb{P}_{\langle Z\subset(3,1,1)\rangle}, \mathbb{P}_{\langle Z\subset(0,0,5)\rangle}$ third case.

FIGURE 4: Comparison of exact and approximate probabilities of being in three representative states across the experiment cases.

performance indicators, such as dependability measures. Assume that the measure of interest is the number of working servers, i.e. those which are not broken. Formally, let E^K denote the probabilistic event that there are K servers running. For instances, when K=5:

$$\mathbb{P}_{\langle Z\subset E^5\rangle} = \mathbb{P}_{\langle Z\subset(5,0,0)\rangle} + \mathbb{P}_{\langle Z\subset(4,1,0)\rangle} + \ldots + \mathbb{P}_{\langle Z\subset(0,5,0)\rangle} \approx$$
$$\mathbb{P}_{\langle Z_a\subset E^5\rangle} = \mathbb{P}_{\langle Z_a=(5,0,0)\rangle} + \mathbb{P}_{\langle Z_a=(4,1,0)\rangle} + \ldots + \mathbb{P}_{\langle Z_a=(0,5,0)\rangle}$$

Using the aggregation method, the steady state values of $\mathbb{P}_{\langle Z\subset E^K\rangle}, K = 1, 2, 5$ and the previously presented outputs were calculated and compared against the corresponding exact results (see Table 1). The comparison provides evidence that a higher probability of being in the model's boundary state corresponds to a less accurate aggregation.

In this experiment, using a quad-core machine with 2G of RAM, for each case, deriving the approximate steady state marginal distribution through the aggregated CTMC took 10–15 seconds. Deriving the same distribution by the exact analysis took 650–700 seconds. The former takes advantage of the possibility of aggregation whereas the latter, derives the marginal distribution from a state space where the detailed dynamics of the clients are also captured. The number of states in the complete state space of the model is 2121 whereas the aggregated one has 21 states. Note that the latter is invariant with respect to the number of clients, highlighting the advantage provided by the aggregation method in terms of the size of the state space.

Applying the aggregation method relies on prior knowledge about the model's behaviour with respect to its boundary states. Such knowledge can be supplied by the domain experts or by monitoring the real system. In large-scale resource-bound communication networks, the resources are almost continuously under contention; e.g. processing input transactions or dealing with a frequent incoming flow of packets. Thus, for such systems, the probability of resources waiting for the users is close to zero and the aggregation algorithm proposed can be safely used.

7. RELATED WORK

Several aggregation and abstraction techniques have been developed to address the problem of state space explosion. One class of methods proposes algorithms aimed at models specified in high level formalisms. The main idea is to develop notions of behavioural equivalence for these languages in terms of their syntax. Then, for a given model, depending on its structure, one may construct one or more abstract models directly from the original model specification. For example, in [5], an aggregation suitable for coloured stochastic Petri nets is presented. In [2], Capra et al. extend this work to stochastic well-formed nets and propose an aggregation exploiting the symmetries present in a model. For models specified in the PRISM language (modules with guarded commands), Dehnert et al. [4] offer a method that uses satisfiability solvers to find bisimilar and minimised Markov chains. For queueing networks, the aggregation presented in [3] takes the description of a multi-class network and under appropriate conditions, generates a number of simpler single-class models. These methods' outputs, i.e. the abstract models, are amenable to analysis at a reduced cost.

An alternative class of aggregation methods works directly on the stochastic process to reduce the state space. Restricting ourselves to Markov chains, these methods develop notions of state equivalence and, when applied on the state space of a large CTMC, they lead to more compact state spaces [1].

The aggregation technique proposed in this paper belongs to the first class; we presented a method suitable for PEPA population models with non-uniform populations, and established its connection with the corresponding low level state space aggregation.

With respect to the aggregation methods suitable for PEPA models, our aggregation method is closely related to two existing concepts; quasi-separability and lumpability. The method of quasi-separability was developed for Markovian process algebra models in [11]. Given a process, the modeller categorises each component as being either an *environment* component or an *internal* one. The environment components are assumed to control the model's high level mode of operation and the internal components capture the internal state. If the model satisfies a *quasi-separability condition*, a number of sub-models are generated where each sub-model captures the dynamics of a number of internal components in combination with all environment ones. For internal components, their configuration is independent of the other internal ones and therefore, their behaviour can be studied *in separation* via only their own sub-model. Using this method for a given model, for each sub-model one marginal distribution is derived. We investigated the quasi-separability condition in the context of PEPA population

	First Case			Second Case			Third Case		
	exact	approximate	error (%)	exact	approximate	error	exact	approximate	error(%)
$\mathbb{P}_{\langle Z_a=500\rangle}$	0.011	0.011	0	0.015	0.011	26	0.016	0.011	31
$\mathbb{P}_{\langle Z_a=311\rangle}$	0.056	0.056	0	0.038	0.056	47	0.020	0.056	180
$\mathbb{P}_{\langle Z_a=005\rangle}$	0.034	0.034	0	0.035	0.034	2	0.039	0.034	12
$\mathbb{P}_{\langle Z_a\subset E^5\rangle}$	0.028	0.028	0	0.023	0.028	21	0.02	0.028	40
$\mathbb{P}_{\langle Z_a\subset E^2\rangle}$	0.310	0.317	2	0.328	0.317	3	0.336	0.317	5.6
$\mathbb{P}_{\langle Z_a\subset E^1\rangle}$	0.161	0.165	2.48	0.171	0.165	3.5	0.189	0.165	12

TABLE 1: Comparion of the equilibrium probabilities calculated separately by the approximate and exact methods.

models. In particular, for a model in which there is an interaction between small and large groups, we established the conditions which determine whether the small groups evolve independently from the large ones and if their dynamics can be studied in isolation.

The second concept is lumpability (ordinary lumpability), which forms a partition over the model's CTMC satisfying a condition on rates [7]. Using lumpability, in the model's aggregated CTMC one state is constructed for each partition. The size of the aggregated CTMC is smaller than the original one and therefore, it provides the means for efficient performance analysis. Our method follows a similar approach. However, our conditions are less stringent than those for lumpability, and will aggregate a larger class of models. More specifically, the presence of boundary states in the CTMC of a model may break the lumpability condition whereas our method can handle such a CTMC.

8. CONCLUSION

In this paper we considered a sub-class of large-scale PEPA models. These models consist of groups with non-uniform population sizes; one or more small groups cooperating with some large ones. These models reflect the situation in many of existing systems where a small population of resources are used by a significantly larger population of users. We showed that if the model satisfies two conditions, an approximate aggregated CTMC can be constructed and used to efficiently derive certain performance metrics. These are the conditions. First, the rate of all cooperation between the model's small groups and large ones should be controlled solely by the small groups. Second, as the model executes, it must be unlikely to experience its boundary states, i.e. the ones where the small groups are blocked while waiting for the cooperation with the large groups. We proposed an algorithm which, for a conforming model, directly derives its aggregated CTMC without constructing the original CTMC. The aggregated CTMC captures, in a detailed manner, the dynamics of the model's small groups and can be used to derive an approximate marginal probability distribution over those groups. Calculating such a distribution using the aggregated CTMC is orders of magnitude faster than deriving the same distribution from the analysis of the original CTMC.

Future Work. For a conforming model, we showed how its aggregated CTMC can be used to study the behaviour of its small groups. However, the aggregated CTMC, as a compact representative of the model's original CTMC, can be exploited more widely. The original CTMC and the aggregated CTMC can be shown to share certain characteristics, yet investigating them in the original CTMC may be prohibitively expensive. Therefore, our aim is to investigate how to establish important characteristics, such as near complete decomposability in the aggregated CTMC and then exploit them in the original CTMC.

9. ACKNOWLEDGMENTS

Jane Hillston is partially supported by the EU project QUANTICOL, 600708.

10. REFERENCES

[1] P. Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of Applied Probability*, 31(1):pp. 59–75, 1994.

[2] L. Capra, C. Dutheillet, G. Franceschinis, and J.-M. Ilie. Exploiting partial symmetries for markov chain aggregation. *Theor. Comp. Sci.*, 39(3):231 – 257, 2000.

[3] A. Conway and N. Georganas. Decomposition and aggregation by class in closed queueing networks. *Software Engineering*, pages 1025–1040, 1986.

[4] C. Dehnert, J.-P. Katoen, and D. Parker. SMT-based bisimulation minimisation of Markov models. In *14th Int. Conf. VMCAI*, pages 28–47, 2013.

[5] C. Dutheillet and S. Haddad. Aggregation of states in colored stochastic petri nets: application to a multiprocessor architecture. In *Proc. of 3rd Int. Workshop PNPM*, pages 40–49, 1989.

[6] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a markovian process algebra. *Theor. Comput. Sci.*, 411(22-24):2260–2297, May 2010.

[7] J. Hillston. *A compositional approach to performance modelling*. CUP, New York, NY, USA, 1996.

[8] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: verification of probabilistic real-time systems. In *23rd Int. Conf.*, CAV, pages 585–591. Springer, 2011.

[9] A. Pourranjbar and J. Hillston. An aggregation technique for large-scale PEPA models with non-uniform populations. Technical report, http://arxiv.org/abs/1309.1613, August 2013.

[10] A. Pourranjbar, J. Hillston, and L. Bortolussi. Don't just go with the flow: Cautionary tales of fluid flow approximation. In *Computer Performance Engineering*, LNCS 7587, pages 156–171. 2013.

[11] N. Thomas and S. Gilmore. Applying quasi-separability to Markovian process algebra. In *PAPM*, 1998.

[12] M. Tribastone. *Scalable Analysis of Stochastic Process Algebra Models*. PhD thesis, School of Informatics, The University of Edinburgh, 2010.