Product Form Solution for a Class of PEPA Models

Jane Hillston[1]

Laboratory for Foundations of Computer Science, University of Edinburgh, UK

Nigel Thomas

Department of Computer Science, University of Durham, UK

Abstract

The advantages of the compositional structure within the Markovian process algebra PEPA for model construction and simplification have already been demonstrated. In this paper we show that for some PEPA models this structure may also be used to advantage during the solution of the model. Several papers offering product form solutions of stochastic Petri nets have been published during the last ten years. In [3], Boucherie showed that these solutions were a special case of a simple exclusion mechanism for the product process of a collection of Markov chains. The results presented in this paper take advantage of his observation. In particular we show that PEPA models that generate such processes may be readily identified and show how the product form solution may be obtained. Although developed here in the context of PEPA the results presented can be easily generalised to any of the other Markovian process algebra languages.

Key words: Markovian process algebra; Product form solutions

1 Introduction

The use of structured modelling paradigms for performance modelling, based on Markov processes, has been advocated by several authors for many years, and several such approaches have been developed and exploited [4,7,24]. Most of these approaches stress compositionality, i.e. the system is viewed as a collection of subsystems that are smaller and more easily modelled. In particular

[1] Corresponding author email: *jeh@dcs.ed.ac.uk*

several Markovian process algebras (MPAs) have been presented in recent years. These include PEPA [8,14], MTIPP [10], and EMPA [1]. Initially the benefits of the compositional structure within these languages have been investigated for model construction and model simplification [9,15,19,23]. However the real strength of this approach will result from the exploitation of this structure to aid model solution.

A variety of decompositional or structural techniques have been proposed to aid in the solution of large Markov processes. Recently several preliminary results have been published which show that, at least for some particular cases, there is a clear relationship between these decompositional techniques and the MPA model descriptions. For example, [21] presents a thorough investigation of the application of time scale decomposition and throughput approximation techniques to MPA models. In [17] a survey of recent work on decomposed solution of MPA models is presented, focusing in particular on models which give rise to product form equilibrium distributions.

In this paper we present an alternative way to exploit the compositional structure of stochastic process algebra models during solution. The previous work on product form MPA models has centred on components of a particular structure which interact in a restricted way, preserving a form of independence between the components. Here we characterise models which represent the competition of otherwise independent processes over resources and identify the cases in which these models exhibit a product form solution. This facilitates a solution technique in which components of the model are solved in isolation, these partial solutions subsequently being combined to give a solution of the complete model. These results are presented in the context of PEPA but can easily be generalised to any of the other Markovian process algebra languages.

The models which we consider here belong to the class of competing Markov processes identified by Boucherie in [3]. The advantage of characterising this class of models in PEPA is that by elevating the definition to a formally defined high-level modelling paradigm we can automatically detect these structures when they occur. The models presented in [3], even those presented as stochastic Petri nets, relied on the insight of the modeller to detect the product form structure. Moreover, in the case of the stochastic Petri net models, a non-standard state representation had to be used in order to eliminate the resource from the model representation. The PEPA models do not have this disadvantage since the resource may (indeed, must) be represented explicitly and subsequently eliminated from the state representation automatically.

The class of models we present here does not extend that introduced by Boucherie. However we hope that the characterisation in PEPA will allow these results to be better understood and to be of greater practical use. More-

over, we aim to ultimately use the class as the target for formally defined (machine-assisted) model simplification techniques.

The remainder of the paper is structured as follows. Section 2 introduces the Markovian process algebra PEPA; Section 3 discusses previous work on product form PEPA models and introduces Boucherie's framework of competing Markov processes. In the following section we apply the framework to PEPA models and so characterise a class of models with product form solution. These results are illustrated by the examples presented in Section 5. To conclude, Section 6 summarises the results of the paper and identifies topics for further work.

2 PEPA

2.1 Introduction

Process algebras are mathematical theories which model concurrent systems by their algebra and provide apparatus for reasoning about the structure and behaviour of the model. In classical process algebras, such as CCS [22], time is abstracted away—actions are assumed to be instantaneous and only relative ordering is represented—and choices are generally nondeterministic. If an exponentially distributed random variable is used to specify the duration of each action the process algebra may be used to represent a Markov process. This is the approach taken by stochastic process algebras such as PEPA. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC) for analytic or numerical solution.

In PEPA a system is described as an interaction of *components* and these components engage, either singly or multiply, in *activities*. The components will correspond to identifiable parts in the system, or roles in the behaviour of the system. They represent the active units within the system; the activities capture the actions of those units. These basic elements of PEPA, components and activities, correspond to *states* and *transitions* in the underlying Markov process. Each activity has an *action type* (or simply *type*). Activities which are private to the component in which they occur are represented by the distinguished action type, τ. The duration of each activity is represented by the parameter of the associated exponential distribution: the *activity rate* (or simply *rate*) of the activity. This parameter may be any positive real number, or the distinguished symbol $\top$ (read as *unspecified*). Thus each activity, a, is a pair (α, r) where α is the action type and r is the activity rate.

We assume that there is a countable set of components, which we denote $\mathcal{C}$, and

a countable set, $\mathcal{A}$, of all possible action types. We denote by $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$, the set of activities, where $\mathbb{R}^+$ is the set of positive real numbers together with the symbol $\top$.

2.2 Syntax and Informal Semantics

PEPA provides a small set of combinators. These allow expressions, or terms, to be constructed defining the behaviour of components, via the activities they undertake and the interactions between them. The combinators, together with their names and interpretations, are presented informally below.

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

Choice: $P + Q$ The component represents a system which may behave either as component P or as Q: all the current activities of both components are enabled. The first activity to complete, determined by a *race condition*, distinguishes one component, the other is discarded. The choice combinator represents competition between components.

Cooperation: $P \underset{L}{\bowtie} Q$ The components proceed independently with any activities whose types do not occur in the *cooperation set* L (*individual activities*). However, activities with action types in the set L require the simultaneous involvement of both components (*shared activities*). These activities are only enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q. Thus one component may become blocked, waiting for the other component to be ready to participate. The cooperation combinator associates to the left but brackets may also be used to clarify the meaning. When the set L is empty, we use the more concise notation $P \parallel Q$ to represent $P \underset{\emptyset}{\bowtie} Q$.

The published stochastic process algebras differ on how the rate of shared activities are defined [13]. In PEPA the shared activity occurs at the rate of the slowest participant (see Appendix A for more details). If an activity has an unspecified rate in a component, the component is *passive* with respect to that action type. This means that the component does not influence the rate at which any shared activity occurs. A model which contains a passive activity without a partner for cooperation is considered to be *incomplete*.

Hiding: P/L The component behaves as P except that any activities of types within the set L are *hidden*, i.e. such an activity exhibits the unknown type τ and the activity can be regarded as an internal delay by the component. Such an activity cannot be carried out in cooperation with any other component: the original action type of a hidden activity is no longer externally

accessible, to an observer or to another component; the duration is unaffected.

Constant: $\boldsymbol{A \stackrel{def}{=} P}$ Constants are components whose meaning is given by a defining equation: $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P. This is how we assign names to components (behaviours). There is no explicit recursion operator but components of infinite behaviour may be readily described using sets of mutually recursive defining equations.

Various forms of equivalence can be defined for the language, and some of them have been shown to be useful for model aggregation [14,15]; however in this paper we will only have recourse to appeal to the simplest, *syntactic equivalence*, denoted $\equiv$.

2.3 Execution strategy

The evolution of a model is governed by the structured operational semantics rules of the language (shown in Appendix A). These define the admissible transitions or state changes associated with each combinator. If a component P can perform an activity (α, r) and become component P' we write $P \xrightarrow{(\alpha,r)} P'$; if P can perform a sequence of activities, $\sigma = a_1 a_2 \cdots a_n$ and evolve into P'' then we write $P \xrightarrow{\sigma} P''$.

A race condition governs the dynamic behaviour of a model whenever more than one activity is enabled. This has the effect of replacing the non-deterministic branching of classical process algebra with probabilistic branching. The probability that a particular activity completes is given by the ratio of its activity rate to the sum of the activity rates of all the enabled activities. Any other activities which were simultaneously enabled will be interrupted or aborted. The memoryless property of the exponential distribution makes it unnecessary to record the remaining lifetime in either case.

The action types which the component P may next engage in are the *current action types* of P, a set denoted $\mathcal{A}(P)$. This set is defined inductively over the syntactic constructs of the language (see [14] for a formal definition). For example, $\mathcal{A}(P + Q) = \mathcal{A}(P) \cup \mathcal{A}(Q)$.

The activities which the component P may next engage in are the *current activities* of P, a multiset denoted $\mathcal{A}ct(P)$. When the system is behaving as component P these are the activities which are enabled. Note that the dynamic behaviour of a component depends on the number of instances of each enabled activity and therefore we consider *multisets* of activities as opposed to *sets* of action types. For any component P, the multiset $\mathcal{A}ct(P)$ is defined inductively over the structure of P, as for $\mathcal{A}(P)$ (see [14] for a formal definition).

The "states" of a PEPA model as it evolves are the syntactic terms, or *derivatives*, which the model will go through. The *derivative set* of a PEPA component C is denoted $ds(C)$ and is the set of components which capture all the reachable states of the system.

It is also necessary to refer to the complete set of action types which are used within the complete behaviour of a component C, i.e. all the possible action types which may be witnessed as a component evolves. This set will be denoted $\vec{\mathcal{A}}(C)$. The *complete action type set* of a component C is

$$\vec{\mathcal{A}}(C) = \bigcup_{C' \in ds(C)} \mathcal{A}(C').$$

Two components, C_1 and C_2, are said to be *distinct* if they have no action types in common, i.e. $\vec{\mathcal{A}}(C_1) \cap \vec{\mathcal{A}}(C_2) = \emptyset$.

2.4 Cyclic PEPA

Unlike the earlier papers [11,18,25], the aim of this paper is to consider models which exhibit a product form solution over the components of the model, even though it may be necessary to find the steady state solution of those components by numerical solution. Thus it is important that we ensure that the components within the model, as well as the model itself, are finite and ergodic. Necessary (but not sufficient) conditions for the ergodicity of the Markov process in terms of the structure of the PEPA model have been identified and can be readily checked [14]. These conditions imply that the model must be a *cyclic* PEPA component.

A cyclic component is one in which behaviour may always be repeated—however the model evolves from this component it will always eventually return to this component and this set of behaviours. In particular for every choice, whichever alternative is chosen the model must eventually return to the point where the choice can be made again, possibly with a different outcome. Since syntactic terms are associated with states, if we consider the layering imposed on a component by cooperation combinators, choice combinators may only be introduced at the lowest level of a cyclic component. In other words, a component which involves a choice combinator may subsequently be used in a cooperation, but a component involving a cooperation may not be subsequently used in a choice. Thus if

$$P \stackrel{def}{=} (\alpha, r).P + (\beta, s).P \qquad S \stackrel{def}{=} (\alpha, r).S + (\beta, s).S$$

$$Q \stackrel{def}{=} (\alpha, r).Q + (\gamma, t).Q \qquad R \stackrel{def}{=} (\beta, s).R + (\gamma, t).R$$

the component $((\alpha, r).P + (\beta, s).P) \underset{\{\alpha\}}{\bowtie} ((\alpha, r).Q + (\gamma, t).Q)$ is a cyclic component but the component $(P \underset{\{\alpha\}}{\bowtie} Q) + (R \underset{\{\beta\}}{\bowtie} S)$ is not.

This leads us to formally define the syntax of PEPA expressions in terms of *sequential components* S and *model components* P:

$$P ::= P \underset{L}{\bowtie} P \mid P/L \mid A$$
$$S ::= (\alpha, r).S \mid S + S \mid A_S$$

where A denotes a constant which is either a sequential or a model component and A_S denotes a constant which is a sequential component. For the remainder of the paper we will assume that all the models which we consider are cyclic.

When a model component is defined it consists of one or more cooperating components, and these cooperating components remain apparent in every derivative of the model. Thus the sequential components involved in the model, and the cooperation sets in operation between them, will remain static throughout its evolution. Only the particular derivatives exhibited by each of the sequential components may change.

Let P be a model component comprising sequential components $S_1, S_2, \ldots S_K$. Then a *state vector* of the model component P as derivative P_i is the vector $(S_{1_i}, S_{2_i}, \ldots, S_{K_i})_P$ where $S_{k_i}, 1 \leq k \leq K$ is the current derivative of S_k in P_i. This can be regarded as analogous to the state representation of a queueing network which consists of a vector $(n_1, n_2, \ldots, n_K)$, where n_i denotes the number of customers currently at queue i. The subscript P is required since knowledge of the static structure of P is needed in order to reason about the model's behaviour. However, it will be omitted when it is clear from the context which P is intended.

A sequential component S_k is *redundant within the state vector representation* of a model component P if S_k is a sequential component of P and for all derivatives $P_i \in ds(P)$ given the current derivatives of the other sequential components $S_{j_i}, j \neq k$, the current derivative of S_k, S_{k_i}, can be inferred. If a sequential component is shown to be redundant within the state vector, a *reduced state vector* may be formed in which the derivatives of this component have been eliminated. A sequential component S_k is *redundant within a model component* P if S_k is a sequential component of P and the Markov process generated by P is isomorphic to the Markov process generated by P', where P' is the model obtained by eliminating S_k from P.

Note that redundancy within the state vector representation is distinct from redundancy within the model. A component which is redundant within the state vector representation may be essential for correct behaviour of the model, imposing some form of scheduling between the other components. In contrast

a component which is redundant within the model has no effect on the behaviour of the model at all. For the remainder of the paper we will use the term *redundant* to refer to components redundant within the state vector representation.

3 Product Form Solutions

3.1 Introduction

Markovian process algebras impose a formally-defined compositional structure on the underlying Markov chain and the exploitation of this structure for model simplification has already been demonstrated. In [15] it is shown that components of a model can be considered and simplified in isolation, thereby avoiding the computational effort required to consider the model as a whole. This technique can be applied to models which exceed the capabilities of the existing techniques. Even greater advantage is gained when the compositional structure can be used during model solution, i.e. if the CTMCs corresponding to the components could be solved separately and their solutions combined to obtain a solution, exact or approximate, of the whole CTMC. One class of CTMCs which are susceptible to such an efficient solution technique are those which exhibit a *product form* equilibrium distribution.

Consider a Markov process $X(t)$, whose state space $\mathcal{S}$ is of the form $\mathcal{S} = S_1 \times S_2$, i.e. each state $\boldsymbol{s} = (s_1, s_2)$ contains two pieces of information capturing different aspects of the current state. In general, these aspects may be related in many ways. When the process $X(t)$ exhibits a product form solution, i.e. $\pi(\boldsymbol{s}) = \pi_1(s_1) \times \pi_2(s_2)$, it indicates that these different aspects of the state description are independent with respect to their equilibrium distribution.

Work on finding MPA models which give rise to product form solutions has drawn on the previous work on both queueing networks and stochastic Petri nets (SPNs). Essentially this can be seen as an investigation of when components interact and yet remain statistically independent. It is clear that when a PEPA model consists of completely independent sequential components, e.g. $C \stackrel{def}{=} P \parallel Q$, the equilibrium distribution over the state vector representation will have a product form:

$$\pi(C_i) = \pi_P(P_j) \times \pi_Q(Q_k) \tag{1}$$

where $C_i \equiv P_j \parallel Q_k$, π_P and π_Q are the equilibrium distributions over the derivatives of P and Q respectively.

In [18] the case of PEPA models exhibiting reversible structures is considered. Informally, a reversible Markov process is one which behaves identically when we observe it with time reversed as when we observe it with time flowing forward. An initial study of the structure of the state space of MPA models giving rise to reversible Markov processes was presented by Bhabuta *et al.* in [2]. Hillston and Thomas [18] identify syntactic conditions which a PEPA model must satisfy in order for the underlying process to be reversible. The problem is tackled in two stages. First, a basic class of sequential components which give rise to reversible structures are identified. Then, assuming that a known class of reversible PEPA components exist, the authors investigate under what circumstances the conditions for reversibility will be preserved if reversible components are composed using the combinators of PEPA.

In [11] an alternative class of models is found based on the notion of quasi-reversibility. A weak form of interaction between components is allowed but components are again restricted to have a particular form. As in the work on reversibility, the approach is to first find simple instances of PEPA processes which give rise to quasi-reversible structure in their underlying Markov process. Then, conditions are established under which these components can be composed while maintaining the quasi-reversible property. In [25] the application of earlier results from a routing process approach in SPN is explored within an MPA setting. The SPN results rely on defining a Markov chain whose states correspond to the transitions of the SPN, the so-called routing chain. The condition for this chain to exist is that the set of places into which tokens are placed when a transition fires should be exactly the input places of another transition [12]. The class of components which may be used within product form models is found to be restricted, although these restrictions are now expressed in terms of the actions of the model and how they are distributed within the components.

In this paper we aim to identify cases when the CTMC underlying a PEPA model has a product form equilibrium distribution and there is no restriction over the possible form of the components which behave independently. In these cases the probability of a given model derivative will be the product of the probabilities of the corresponding derivatives in the quasi-independent components, possibly subject to a normalisation constant. The approach taken here is analogous to the approach used by Lazar and Robertazzi with SPN [20], since the aim is to find a product form in terms of submodels (subnets or components respectively). CTMCs may still need to be solved numerically to find the equilibrium distribution of the submodels but these Markov processes will be smaller and can be tackled separately. Unlike the other approaches, we maintain the restriction that the subcomponents we consider in the state representation are independent of each other. However, we extend from the simple case represented in equation (1) above by the introduction of a redundant resource which imposes indirect interaction between the components, but

which may be eliminated from the state vector representation.

3.2 Boucherie's Framework

In [3], Boucherie generalises the result of Lazar and Robertazzi [20] to show that it can be regarded as a special case of a simple exclusion mechanism for the product process of a collection of Markov processes. His framework consists of a set of Markov processes which must compete over resources. This competition means that there are certain areas of the state space of the product process which cannot be entered—these areas would correspond to two processes holding a resource at the same time. He identifies circumstances in which, despite this indirect form of interaction, the product process exhibits a product form equilibrium distribution over the permissible states, suitably renormalised. In this subsection we summarise Boucherie's framework: our presentation closely follows his and for consistency we adopt his notation.

Let S_k and q_k be the state space and transition rates, respectively, of the kth Markov process in a collection, $1 \leq k \leq K$. Let $\mathcal{S} = S_1 \times \cdots \times S_K$ denote the state space of the *product process*. It is assumed that in each transition of the product process only the state in one dimension changes, i.e. in each transition of the product process only one of the underlying Markov processes changes its state. The transition rates of the product process in dimension k are then given by the transition rates of the individual process, q_k. In other words, the product process consists of K individual processes which do not directly interact in any way.

Competition between the processes over resources introduces an indirect form of interaction between them. Let us assume that there is a set I of *notional resources*. For technical reasons this set may also contain a non-resource, possession of which will indicate that a process is working without a resource. Then the state space of each Markov process can be partitioned into states corresponding to resource use. Only one notional resource may be held at a time but this does not imply that only one actual resource is held since different notional resources may correspond to a resource held individually, and a resource held in conjunction with another. Competition over resources is then defined as follows:

Definition 1 (Competition)
Let I be an index set. For each k, let $A_{ki}, i \in I$, be a partition over S_k such that $\emptyset \neq A_{ki} \subset S_k$ and $\bigcup_{i \in I} A_{ki} = S_k$, $k = 1, \ldots, K$. Markov process k uses resource i if the Markov process is in state $n_k \in A_{ki}$. Markov processes k_1 and k_2 compete over resource i if $\{n_{k_1}, n_{k_2} : n_{k_1} \in A_{k_1 i}, n_{k_2} \in A_{k_2 i}\} = \emptyset$. Let $C_{ki} \subset \{1, \ldots, K\}$ be the Markov processes that compete over resource i with

Markov process k.

For any process k, $1 \leq k \leq K$, if the current state is in the subset A_{ki} it signifies that the process is presently using the resource i and no other process j, such that $j \in C_{ki}$, can gain access to i, entering its subset of states A_{ji}. Thus the competition and the sets C_{ki} define areas of the state space of the product process which are inaccessible. The transition rates of the product process are defined in a way which ensures this exclusion.

Definition 2 (Product process)
The Markov process at state space $\mathcal{S} = \prod_{k=1}^{K} S_k$, with transition rates

$$q(n, n') = \sum_{k=1}^{K} q_k(n_k, n'_k) \prod_{\ell=1, \ell \neq k}^{K} \mathbf{1}(n_\ell = n'_\ell) \mathbf{1}(\forall i, n_\ell \in A_{\ell i} \rightarrow k \notin C_{\ell i})$$

where $n = (n_1, \ldots, n_K)$, $n' = (n'_1, \ldots, n'_K)$, is called the product process *of the collection of Markov processes $1, \ldots, K$, competing over resources I.*

The indicator function $\mathbf{1}$ is such that the value of $\mathbf{1}(\textit{statement})$ is 1 if the statement is true and 0 otherwise. Thus, at each transition, the product process selects a transition from a single constituent process and proceeds at the rate of that transition.

Observe that, as required, these transition rates imply that in each transition only one process can change its state, and that process k cannot progress if resource i is being used by process ℓ and $k \in C_{\ell i}$, and vice versa. In his paper Boucherie shows that this mechanism of pair-wise relations imposed on the individual processes of a product process can be used to model various types of competition.

Theorem 3 (Product-form distribution)
The product process of the collection of Markov processes $1, \ldots K$ competing over resources I has equilibrium distribution π at $\mathcal{S}$, defined as

$$\mathcal{S} = \prod_{k=1}^{K} S_k \setminus \left(\prod_{k=1}^{K} \prod_{i \in I} \prod_{j \in C_{ki}} A_{ki} \times A_{ji} \right)$$

given by

$$\pi(n) = B \prod_{k=1}^{K} \pi_k(n_k) \qquad n \in \mathcal{S} \tag{2}$$

where B is a normalising constant, determined by the exact form of $\mathcal{S}$, and $\pi_k(\cdot)$ is the equilibrium distribution of process S_k.

The result holds because each process can either operate independently of the

other processes or it is blocked. For all $n \in \mathcal{S}$, if process ℓ is in state n_ℓ and $\ell \neq k$ then process k either carries out a transition independently because it is not in competition with ℓ ($\mathbf{1}(\forall i, n_\ell \in A_{\ell i} \rightarrow k \notin C_{\ell i}) = 1$) or process k is blocked because it needs to access the resource which ℓ occupies ($\mathbf{1}(\forall i, n_\ell \in A_{\ell i} \rightarrow k \notin C_{\ell i}) = 0$). In either case process k will satisfy its own global balance equations: these equations are trivially satisfied when the process is stopped and are also true when the process is operating independently. It appears that the exclusion principle incorporated in the transition rates of the product process amounts to a redefinition of the product process such that the Markov processes in the collection operate as if they are independent.

Boucherie shows that the theorem can be generalised if we consider the case when each of the processes $1, \ldots, K$ is composed of several locally balanced Markov processes. Assume that the transition rates of Markov process k can be separated into T_k parts, labelled $t = 1, \ldots, T_k$. For $n_k, n'_k \in S_k$ we define the separated transition rates as $q_k^t(n_k, n'_k)$, $t = 1, \ldots T_k$, such that

$$q_k(n_k, n'_k) = \sum_{t=1}^{T_k} q_k^t(n_k, n'_k) \tag{3}$$

The Markov process k is said to be *locally balanced* with respect to this separation if the equilibrium distribution π_k satisfies

$$\sum_{n'_k \in S_k} \left\{ \pi_k(n_k) q_k^t(n_k, n'_k) - \pi_k(n'_k) q_k^t(n'_k, n_k) \right\} = 0 \qquad t = 1, \ldots, T_k.$$

Each process (k, t), $1 \leq k \leq K$, $1 \leq t \leq T_k$, is a Markov process in its own right. Thus for each k there may be several subprocesses $(k, t), 1 \leq t \leq T_k$, which all operate over the same state space S_k. Note that they do not necessarily all use the whole state space. We assume that for any k, subprocesses (k, t_i) and (k, t_j) do not compete over any resources. However, we extend the notion of competition between processes in the natural way: for each (k, t), let $A_{(k,t)i}$ be mutually exclusive sets such that $A_{(k,t)i} \subset S_k$, $\bigcup_{i \in I} A_{(k,t)i} \subseteq S_k$ and $\bigcup_{t=1}^{T_k} \bigcup_{i \in I} A_{(k,t)i} = S_k$. Similarly, let $C_{(k,t)i} \subseteq \{(k, t) : 1 \leq k \leq K, 1 \leq t \leq T_k\}$ be the set of Markov processes which compete with process (k, t) over resource i, i.e. if $(k', t') \in C_{(k,t)i}$ then $\{n_k \in A_{(k,t)i}, n'_k \in A_{(k',t')i}\} = \emptyset$.

Theorem 4 (Product process with local balance)
Assume that each Markov process in the collection $k = 1, \ldots, K$ satisfies local balance with respect to the separation (3). The Markov process at state space

$$\mathcal{S} = \prod_{k=1}^{K} S_k \setminus \left(\prod_{k=1}^{K} \prod_{i \in I} \prod_{t=1}^{T_k} \prod_{(j,s) \in C_{(k,t)i}} A_{(k,t)i} \times A_{(j,s)i} \right)$$

with transition rates

$$q(n,n') = \sum_{k=1}^{K} \sum_{t=1}^{T_k} q_k^t(n_k, n_k') \prod_{\ell=1,\ell\neq k}^{K} \mathbf{1}(n_\ell = n_\ell') \prod_{p=1}^{T_\ell} \mathbf{1}(\forall i, n_\ell \in A_{(\ell,p)i} \rightarrow (k,t) \notin C_{(\ell,p)i})$$

has an equilibrium distribution π *at* $\mathcal{S}$ *given by*

$$\pi(n) = B \prod_{k=1}^{K} \pi_k(n_k)$$

where B *is a normalising constant.*

4 PEPA Components Competing over Resources

In order to apply Boucherie's framework in PEPA we need to clarify the notions associated with a resource. When a model component is constructed from sequential components via the cooperation combinator we can regard the model as being built up in layers or levels, each cooperation combining just two components. As suggested by the grammar, those components may be sequential components or model components. Thus a sequential component may be within the scope of several cooperation sets because of the way the model has been constructed.

It is clearly useful to be able to establish which cooperation sets will affect the behaviour of the component. For example, in the component

$$X \stackrel{def}{=} (P \underset{L}{\bowtie} R) \underset{K}{\bowtie} (S \underset{N}{\bowtie} T)$$

the subcomponent R can act independently on any action types in the set N which do not occur in K or L, but must have the cooperation of other subcomponents to achieve actions in the set $L \cup K$, whereas the subcomponent S can act independently on any action types in the set $L \setminus (K \cup N)$, but must have the cooperation of other subcomponents to achieve actions in the set $K \cup N$.

In the following we formalise this idea. First, we define a partial order, $\prec$, over components, which captures the notion of *being a subcomponent*:

Definition 5 (Subcomponents)

(1) $R \prec P$ *if* $R \in ds(P)$
(2) $R \prec P + Q$ *if* $R \prec P \vee R \prec Q$
(3) $R \prec P \underset{L}{\bowtie} Q$ *if* $R \prec P \vee R \prec Q$
(4) $R \prec P/L$ *if* $R \prec P$

(5) $R \prec A \qquad$ *if* $A \stackrel{\text{def}}{=} P \wedge R \prec P$

The *interface* of a sequential component within a component model is defined to be the union of all the cooperation sets whose scope includes the component R.

Definition 6 (Interface)
For any sequential component R within a model component C (i.e. $R \prec C$) the interface *of R within C, denoted $\mathcal{I}(C :: R)$, is the set of action types on which R is required to cooperate. It is defined in terms of the subsidiary function $\mathcal{J}$; $\mathcal{I}(P :: R) = \mathcal{J}(P :: R, \emptyset)$, where $\mathcal{J}$ is defined as follows*

(1) $\mathcal{J}(R :: R, K) = K$

(2) $\mathcal{J}(P \underset{L}{\bowtie} Q :: R, K) = K' \cup K''$ *if* $\mathcal{J}(P :: R, K \cup L) = K'$ *and* $\mathcal{J}(Q :: R, K \cup L) = K''$

(3) $\mathcal{J}(P/L :: R, K) = \mathcal{J}(P :: R, K \setminus L)$

(4) $\mathcal{J}(A :: R, K) = \mathcal{J}(P :: R, K)$ *if* $A \stackrel{\text{def}}{=} P$

(5) $\mathcal{J}(R' :: R, K) = \emptyset$ *if* $R \not\prec R'$.

When all the possible actions of a sequential component are constrained by its interface the component is never free to act independently; it must cooperate with other components to complete *any* action. Such a component can be viewed as being subservient to the rest of the model, and is called a *resource component.*

Definition 7 (Resource Components)
A sequential component R in a model C is a resource component *if there is only one instance of R within C and the complete action type set of R is a subset of its interface within C, i.e.*

$$\vec{\mathcal{A}}(R) \subseteq \mathcal{I}(C :: R)$$

A resource component is clearly closely related to the notion of redundancy. It is only possible for a sequential component to be redundant within a model while not being a resource component if independent actions by the component do not result in a change of derivative.

A sequential component S is termed a *basic component* if it is a cyclic component constructed using only prefix and each activity in the cycle is of a distinct type. A sequential component R is termed a *simple component* if

$$R \equiv S_1 + S_2 + \cdots + S_n$$

for some distinct basic components $S_1, S_2, \ldots S_n$, modified so that the last action of each loop returns to R and not S_i. If a resource component is simple it implies that it offers alternative behaviours through its interface but once

one of those behaviours is chosen (on the first action) the pattern of behaviour is set until the chosen cycle is completed and the choice is offered again.

To fit within Boucherie's framework we need to impose some restrictions on the syntactic form of the resource component, and its interaction with the rest of the model, in order to ensure that the exclusion property will hold.

Definition 8 (Guarding Resource)
A simple resource component, R, in the model component $S \underset{L}{\bowtie} R$ is a guarding resource *with respect to the sequential component S if for each α_i such that $S \equiv \sum(\alpha_i, r_i).S_i$ either $\alpha_i \in \mathcal{A}(R) \cap L$ or every activity sequence σ which starts with α_i is such that $S_i \xrightarrow{\sigma} S$ and $\sigma \cap \mathcal{A}(R) = \emptyset$.*

Since S is a sequential component it is constructed using only prefix and choice, which means that in its initial state it either has one behaviour (prefix) or a set of alternative behaviours (choice). If it has a single behaviour, R is a guarding resource if S is required to cooperate with R to start its behaviour, or if the complete behaviour of S is independent of R, i.e. there is no cooperation between them. If S has a set of alternative behaviours represented by a choice of activities in the initial state, then R is a guarding resource if R's cooperation is required for the first activity or, for any complete loop which S can start from this activity, it can return to S without interaction with R. Note that this implies that the Markov processes underlying S can be separated into those requiring interaction with R and those not in such a way that the local balance property holds.

Definition 9 (Returning Resource)
A simple resource component, R, in the model component $S \underset{L}{\bowtie} R$ is a returning resource *with respect to the sequential component S if for each derivative $S' \underset{L}{\bowtie} R'$ such that $S' \equiv (\alpha, r).S$ then either $R' \equiv (\alpha, s).R$ for some s, or $R' \equiv R$ and there is a succession of derivatives from $S \underset{L}{\bowtie} R$ to $S' \underset{L}{\bowtie} R$ in which R does not change state.*

The condition on the returning resource is similar to that of the guarding resource but acting at the end of alternative behaviours instead of the start. For each possible loop of the sequential component, if it cooperates with the resource component at all it must do so on the last activity of the loop.

In order to ensure Boucherie's exclusion mechanism we need resource components which are both guarding and returning resources with respect to the sequential components. We say that a resource R is a *guarding and returning resource* with respect to a sequential component if, for each activity sequence which is guarded by R, R is also a returning resource.

Consider the components

$$P \stackrel{def}{=} (\alpha_1, r_1).(\beta_1, s_1).(\gamma_1, t_1).P + (\alpha_2, r_2).(\beta_2, s_2).(\gamma_2, t_2).P$$

$$\begin{aligned}
R_1 &\stackrel{def}{=} (\alpha_1, r_1).(\beta_1, s_1).R_1 \qquad & L_1 &= \{\alpha_1, \beta_1\} \\
R_2 &\stackrel{def}{=} (\beta_1, s_1).(\beta_1, s_1).R_2 \qquad & L_2 &= \{\beta_1, \gamma_1\} \\
R_3 &\stackrel{def}{=} (\alpha_1, r_1).(\gamma_1, s_1).R_3 \qquad & L_3 &= \{\alpha_1, \gamma_1\} \\
R_4 &\stackrel{def}{=} (\alpha_1, r_1).(\gamma_2, s_2).R_4 \qquad & L_4 &= \{\alpha_1, \gamma_2\}
\end{aligned}$$

It is clear that in each $P \underset{L_i}{\bowtie} R_i$ $(i = 1, 2, 3)$, the component R_i is a resource. However only component R_3 is a guarding and returning resource with respect to P. R_1 is a guarding but not a returning resource, whereas R_2 is a returning but not a guarding resource. R_4 is a guarding resource with respect to the first behaviour of P and a returning resource with respect to the second but this does not make it a guarding and returning resource.

We are now in the position to state the simplest class of PEPA models that satisfy Boucherie's condition.

Theorem 10
Let M be a model component, $M \equiv (S_1 \parallel S_2 \parallel \cdots \parallel S_K) \underset{L}{\bowtie} R$ where each S_i is a sequential component and R is a simple component. Assume that the cooperation set L is such that R is a redundant resource within the state representation vector of M, and moreover, when we consider the model components $S_k \underset{L}{\bowtie} R$ for all $k, 1 \leq k \leq K$, R is a guarding and returning resource with respect to S_k. Then the equilibrium probability of any derivative M' which has state vector representation $(S'_1, \ldots, S'_K, R')$ is given by the product form

$$\pi(S'_1, \ldots, S'_K, R') = B \prod_{k=1}^{K} \pi_k(S'_k \underset{L}{\bowtie} R'_k)$$

where B is the normalising constant, π_k is the equilibrium probability distribution of $S_k \underset{L}{\bowtie} R$ and R'_k is the derivative of R which can be inferred from S'_k, since R is redundant in the state vector representation of $S_k \underset{L}{\bowtie} R$.

The proof of Theorem 10 relies on showing that the Markov process underlying the model M satisfies Boucherie's conditions, i.e. it consists of the product of a number of competing Markov processes, such that only one competing process may change its state at a time and that they observe the exclusion property with respect to the resource. The mapping from the state representation of the PEPA model to the product process is straightforward. We associate one Markov process with each sequential component S_i in the model; the state space of the Markov process is simply the derivative set of the component. The PEPA semantics and the mutual independence of the sequential components

ensures that only one product process will change state in any transition (using the reduced state vector representation). It is then not difficult to show that the competition imposed by the PEPA semantics over the resource component does indeed provide the Boucherie exclusion (see [16]).

The restriction that the resource within the system is comprised of a single sequential resource component is unnecessarily strict. It is straightforward to make the result more general by considering a set of mutually independent resources over which the other components compete.

Corollary 11
Let M be a model component, $M \equiv (S_1 \parallel S_2 \parallel \cdots \parallel S_K) \underset{L}{\bowtie} R$, as in Theorem 10, except that $R \equiv R_1 \parallel \cdots \parallel R_m$, and the R_i are distinct simple components, i.e. $\vec{\mathcal{A}}(R_i) \cap \vec{\mathcal{A}}(R_j) = \emptyset$ for all $i, j, 1 \leq i, j, \leq m$. Then the equilibrium probability of any derivative M' which has state vector representation $(S'_1, \ldots, S'_K, R'_1, \ldots, R'_m)$ is given by the product form

$$\pi(S'_1, \ldots, S'_K, R'_1, \ldots, R'_m) = B \prod_{k=1}^{K} \pi_k(S'_k \underset{L}{\bowtie} (R'_{1k} \parallel \cdots \parallel R'_{mk}))$$

where B is the normalising constant, π_k is the equilibrium probability distribution of $S_k \underset{L}{\bowtie} R$ and R'_{jk} is the derivative of R_j which can be inferred from S'_k.

As previously, it is a straightforward matter to show that if we form the product process based on the S_k then it satisfies the conditions of Theorem 4.

Another generalisation can be achieved if we consider models in which the competing processes are not restricted to be sequential components but may themselves be model components. In order to satisfy the Boucherie conditions these model components must still be mutually independent but each model component may consist of the cooperation of several sequential components. The definitions for guarding and returning resources can be extended from sequential components to model components in a natural way. We require that the resource component is a guarding (returning) resource with respect to each sequential component considered in isolation, and moreover, that if one of a pair of interacting sequential components requires the resource in order to start a behavioural loop then the other must do so also.

Definition 12 (Guarding resource for a model component)
Consider a model component P which consists of n sequential components which cooperate over arbitrary sets of action types:

$$P \equiv S_1 \underset{L_1}{\bowtie} S_2 \underset{L_2}{\bowtie} \cdots \underset{L_{n-1}}{\bowtie} S_n$$

Let R be a redundant resource in the model $P \underset{L}{\bowtie} R$ such that R is a guarding resource with respect to each S_k in the model $S_k \underset{L}{\bowtie} R$. Without loss of

generality assume each S_k has the form $S_k \equiv \sum_{t=1}^{T_k}(\alpha_{kt}, r_{kt}).S_{kt}$. Then R is a guarding resource with respect to P *if whenever, for any k and ℓ there exist t and t' such that $\vec{\mathcal{A}}_{S_k}(S_{kt}) \cap \vec{\mathcal{A}}_{S_\ell}(S_{\ell t'}) \cap \mathcal{I}(P :: S_k) \cap \mathcal{I}(P :: S_\ell) \neq \emptyset$ then $\alpha_{kt} = \alpha_{\ell t'}$ and $\alpha_{kt} \in \mathcal{A}(R) \cap \mathcal{I}(P :: S_k) \cap \mathcal{I}(P :: S_\ell)$.*

Definition 13 (Returning resource for a model component)
Consider a model component P which consists of n sequential components which cooperate over arbitrary sets of action types:

$$P \equiv S_1 \underset{L_1}{\bowtie} S_2 \underset{L_2}{\bowtie} \cdots \underset{L_{n-1}}{\bowtie} S_n$$

Let R be a redundant resource in the model $P \underset{L}{\bowtie} R$ such that R is a returning resource with respect to each S_k in the model $S_k \underset{L}{\bowtie} R$. Without loss of generality assume each S_k has the form $S_k \equiv \sum_{t=1}^{T_k}(\alpha_{kt}, r_{kt}).S_{kt}$. Then R is a returning resource with respect to P *if whenever, for any k and ℓ there exist t and t' such that $\vec{\mathcal{A}}_{S_k}(S_{kt}) \cap \vec{\mathcal{A}}_{S_\ell}(S_{\ell t'}) \cap \mathcal{I}(P :: S_k) \cap \mathcal{I}(P :: S_\ell) \neq \emptyset$ then there is a derivative of S_{kt}, $S'_{kt} \equiv (\beta_{kt}, r_{\beta k}).S_k$ and a derivative of $S_{\ell t'}$, $S_{\ell t'} \equiv (\beta_{\ell t'}, r_{\ell t'}).S_\ell$ such that $\beta_{kt} = \beta_{\ell t'}$ and $\beta_{kt} \in \vec{\mathcal{A}}(R) \cap \mathcal{I}(P :: S_k) \cap \mathcal{I}(P :: S_\ell)$.*

We define R to be a guarding and returning resource as previously. We are now in the position to state a more general theorem:

Theorem 14
Let M be a model component, $M \equiv (P_1 \parallel P_2 \parallel \cdots \parallel P_K) \underset{L}{\bowtie} R$, where R is a parallel composition of distinct simple components. Assume that the cooperation set L is such that each R_i is a redundant resource within the state representation vector of M, and moreover, when we consider the model components $P_k \underset{L}{\bowtie} R$ for all $k, 1 \leq k \leq K$, R is a guarding and returning resource with respect to P_k. Then the equilibrium probability of any derivative M' which has state vector representation $(P'_1, \ldots, P'_K, R')$ is given by the product form

$$\pi(P'_1, \ldots, P'_K, R') = B \prod_{k=1}^{K} \pi_k(P'_k \underset{L}{\bowtie} R'_k)$$

where B is the normalising constant, π_k is the equilibrium probability distribution of $P_k \underset{L}{\bowtie} R$ and R'_k is the derivative of R which can be inferred from P'_k, since R is redundant in the state vector representation of $P_k \underset{L}{\bowtie} R$.

Proof
As previously, we have to show that if we form the product process based on the P_k then it satisfies the conditions of Theorem 4. Assume that each model component P_k is comprised of sequential components $S_{k1}, \ldots, S_{kN_k}$. If each R is a guarding and returning resource with respect to each P_k, it follows that each R_i is a guarding and returning resource with respect to each $S_{kj}, 1 \leq j \leq N_k$. Each simple resource component, R_i, corresponds to a

notional resource, r_i. Thus, the set of notional resources is $I = \{0, r_1, \ldots, r_m\}$ where 0 denotes independent activity without the cooperation of a resource and r_i denotes activity with the cooperation of the resource component R_i.

We now wish to associate each of the model components P_k with one product process in the underlying Markov process. The separation of each process into locally balanced processes is not completely straightforward.

Assume

$$P_k \equiv S_{k1} \underset{L_{k1}}{\bowtie} S_{k2} \underset{L_{k2}}{\bowtie} \cdots \underset{L_{kN_k-1}}{\bowtie} S_{kN_k}$$

where each $S_{kj} \equiv \sum_{t=1}^{T_{kj}} (\alpha_{kjt}, r_{kjt}).S_{kjt}$. If we consider S_{kj} in isolation, since R is a guarding and returning resource with respect to S_{kj} the underlying Markov process can be separated into T_{kj} locally balanced processes. If we consider P_k, since R is a guarding and returning resource with respect to P_k, any pair of sequential components S_{kj} and S_{ki} only interact if they do so on the first activities of loops S_{kjt} and $S_{kit'}$ say. It follows that we can separate the Markov process underlying P_k into locally balanced Markov processes which correspond directly with the enabled activities of P_k, $\mathcal{A}ct(P_k) = \{\!| \alpha_{k1}, \ldots, \alpha_{km} |\!\}$. Thus we can denote the set of locally balanced Markov processes within P_k as $\{S_{P_k}^{(1)}, \ldots, S_{P_k}^{(m)}\}$.

The state space of each $S_{P_k}^{(t)}$ can be partitioned according to the set

$$\{0, r_1, \ldots, r_m\}$$

and the sets of competing processes $C_{(k,t)}$ can be readily identified by considering the initial actions of the S_{kj}. The PEPA semantics ensures that the state space is

$$\mathcal{S} = \prod_{k=1}^{K} ds(P_k) \setminus \left(\prod_{k=1}^{K} \prod_{i \in I} \prod_{t=1}^{T_k} \prod_{(j,s) \in C_{(k,t)i}} A_{(k,t)i} \times A_{(j,s)i} \right)$$

and that the transition rates are

$$q(M', M'') = \sum_{k=1}^{K} \sum_{t=1}^{T_k} q_k^t(P_k', P_k'') \prod_{\ell=1, \ell \neq k}^{K} \mathbf{1}(P_\ell' = P_\ell'') \prod_{p=1}^{T_\ell} \mathbf{1}(\forall i, P_\ell' \in A_{(\ell,p)i} \rightarrow (k,t) \notin C_{(\ell,p)i})$$

Thus, by Theorem 4, we can conclude that the equilibrium distribution π of M' is given by

$$\pi(P_1', \ldots, P_K', R_1', \ldots, R_m') = B \prod_{k=1}^{K} \pi_k(P_k', R_{1k}', \ldots, R_{mk}')$$

where B is a normalising constant. $\square$

5 Examples

In this section we present three examples which illustrate our class of models. The first is taken from Boucherie's paper; the second shows that there is some overlap between our class of models and those identified by Sereno [25]; the third demonstrates an application of Theorem 16.

5.1 Concurrent processing and database locking

We consider the database locking protocol modelled in Boucherie's paper directly as a product process of Markov processes. We show that the system can naturally be modelled using PEPA, generating the same set of competing Markov product processes. Moreover, this can be recognised without recourse to the state space since the PEPA model satisfies the syntactic conditions identified in the previous section.

We assume that the database consists of N items and that each transaction is associated with a list of items in the database—those items needed to process the transaction. Each such list can be partitioned into two sets, with items in the leading part requiring exclusive locks, and items in the trailing set requiring non-exclusive locks. We assume all transactions exclusively lock at least one item. Transaction processing requests arrive from the environment; on arrival, the database lock manager decides whether to grant or refuse the request on the basis of the locks required. Let W_d and R_d be the lists of exclusively locked and non-exclusively locked items respectively in the database at the time of arrival. Let W_a and R_a be the lists of items required, exclusively and non-exclusively, by the arriving request. The locks are granted and the processing request accepted if

$$(W_a \cap W_d) \cup (W_a \cap R_d) \cup (R_a \cap W_d) = \emptyset.$$

Otherwise the locks are denied and the request for transaction processing is blocked, cleared, and discarded. If the locks are granted, and the transaction is accepted for processing, the locks are not released until the entire processing of the transaction is complete.

There are K types of transactions, labelled $k = 1, \ldots, K$. Assume that a transaction of type k requires items $j_k \subset \{1, \ldots, N\}$ and $p_k \subset \{1, \ldots, N\}$ to be exclusively and non-exclusively locked respectively. If transactions of type k arrive at Poisson rate λ_k and are served at exponential rate μ_k, then the

database lock protocol can be modelled as a PEPA process. We associate one PEPA component with each transaction type and each item in the database.

$$\begin{aligned} Trans_k &\stackrel{def}{=} (arrive_k, \lambda_k).(\mathit{finish}_k, \mu_k).Trans_k \\ Item_i &\stackrel{def}{=} \sum_{k \in T(i)} (arrive_k, \top).(\mathit{finish}_k, \top).Item_i \end{aligned}$$

where $T(i) = \{k : i \in j_k\}$, is the set of transactions which require exclusive access to item i.

The complete system is

$$(Trans_1 \parallel \cdots \parallel Trans_K) \underset{L}{\bowtie} (Item_1 \parallel \cdots \parallel Item_N)$$

where $L = \{arrive_k, \mathit{finish}_k \forall k\}$. Each $Item_i$ is a simple guarding and returning resource with respect to the components $Trans_k$, $1 \leq k \leq K$. Following Boucherie, we can see that the equilibrium distribution is

$$\pi(\bar{n}) = B \prod_{k=1}^{K} \left(\frac{\lambda_k}{\mu_k}\right)^{n_k}$$

where B is the normalising constant, and

$$n_k = \begin{cases} 1 & \text{if } Trans'_k \equiv (\mathit{finish}_k, \mu_k).Trans_k \\ 0 & \text{otherwise} \end{cases}$$

5.2 The Taxi Rank System

In [25] a simple taxi rank system is considered. In this system there are a given number of customers and a given number of taxis. If there is an available taxi in the taxi rank a customer has two possibilities. He can take a taxi to the market; the taxi then waits for the customer to complete his shopping and then returns the customer to the taxi rank, becoming available for the next customer. Alternatively the customer can choose to walk through the garden and then go back to the taxi rank. If there are no taxis available at the rank the customer can only go for a walk.

A PEPA model of the customer is as follows:

$$\begin{aligned} C_1 &\stackrel{def}{=} (\mathit{to_market}, r_1).C_2 + (\mathit{to_garden}, r_2).C_3 \\ C_2 &\stackrel{def}{=} (\mathit{from_market}, r_3).C_1 \\ C_3 &\stackrel{def}{=} (\mathit{from_garden}, r_4).C_1 \end{aligned}$$

while the taxi can be represented as:

$$T_1 \stackrel{def}{=} (to_market, \top).T_2$$
$$T_2 \stackrel{def}{=} (from_market, \top).T_1$$

A taxi rank system in which there are two customers and one taxi is modelled by

$$TS \stackrel{def}{=} (C_1 \parallel C_1) \underset{\substack{\{to_market,\\ from_market\}}}{\bowtie} T_1$$

In this model it is trivial to see that the taxi is a guarding and returning resource with respect to each of the customers. Considering a customer in isolation we find that:

$$\pi_C(C_1) = \frac{r_2 r_4}{r_2 r_4 + r_1 r_4 + r_2 r_3}$$
$$\pi_C(C_2) = \frac{r_1 r_4}{r_2 r_4 + r_1 r_4 + r_2 r_3}$$
$$\pi_C(C_3) = \frac{r_2 r_3}{r_2 r_4 + r_1 r_4 + r_2 r_3}$$

Thus, we can deduce that the equilibrium distribution is:

$$\pi(C_i, C_j, T_k) = B\pi_C(C_i) \times \pi_C(C_j)$$

where B is the normalising constant and the $\pi_C(C_i)$ are as above.

The significance of this example is that it shows that there is an intersection between the class of PEPA models which satisfies Sereno's criteria for product form and those which satisfy Boucherie's.

5.3 Geographical Information System

Finally we consider an example in which the competing processes are not sequential components but more complex subsystems. A navigational device consists of a pair of sensors which maintain complementary data about geographical location. In order to keep the sensors' internal data structures in complementary states, they share data via a register. Each sensor gains access to the register and locks it while it reads the current data value; it then uses this information to adjust the equipment it controls while also recalculating a value for the shared register based on its own internal data structures. It then updates the value in the register and releases it. In addition, sensor 1 will periodically gather data and use it to recalculate its internal data structures.

Each sensor can be represented as a pair of sequential components: one respon-

sible for resetting the equipment during each cycle, and one responsible for carrying out the data recalculation. In sensor 1, the recalculation component is assumed to be also responsible for gathering data. Thus the PEPA model of sensor 1 is:

$$
\begin{aligned}
S_{110} &\stackrel{def}{=} (read_data, r_1).S_{111} + (gather_data, r_5).S_{113} \\
S_{111} &\stackrel{def}{=} (recalculate, r_2).S_{112} \\
S_{112} &\stackrel{def}{=} (update_data, r_3).S_{110} \\
S_{113} &\stackrel{def}{=} (recalculate, r_2).S_{110}
\end{aligned}
$$

$$
\begin{aligned}
S_{120} &\stackrel{def}{=} (read_data, r_1).S_{121} \\
S_{121} &\stackrel{def}{=} (reset_apparatus, r_4).S_{122} \\
S_{122} &\stackrel{def}{=} (update_data, r_3).S_{120}
\end{aligned}
$$

$$Sensor_{10} \stackrel{def}{=} S_{110} \underset{L}{\bowtie} S_{120}$$

where $L = \{read_data, update_data\}$.

The second sensor is defined similarly:

$$
\begin{aligned}
S_{210} &\stackrel{def}{=} (read_data, r_1).S_{211} \\
S_{211} &\stackrel{def}{=} (recalculate, r_6).S_{212} \\
S_{212} &\stackrel{def}{=} (update_data, r_3).S_{210}
\end{aligned}
$$

$$
\begin{aligned}
S_{220} &\stackrel{def}{=} (read_data, r_1).S_{221} \\
S_{221} &\stackrel{def}{=} (reset_apparatus, r_7).S_{222} \\
S_{222} &\stackrel{def}{=} (update_data, r_3).S_{220}
\end{aligned}
$$

$$Sensor_{20} \stackrel{def}{=} S_{210} \underset{L}{\bowtie} S_{220}$$

The data register is represented as a simple resource:

$$R \stackrel{def}{=} (read_data, \top).(update_data, \top).R$$

The complete system is represented as:

$$M \stackrel{def}{=} (Sensor_{10} \parallel Sensor_{20}) \underset{L}{\bowtie} R$$

If we consider $Sensor_{10} \underset{L}{\bowtie} R \equiv (S_{110} \underset{L}{\bowtie} S_{120}) \underset{L}{\bowtie} R$ it is straightforward to see that the component R is a guarding and returning resource with respect to $Sensor_{10}$: S_{110} and S_{120} cooperate on the actions *read_data* and *update_data* and $Sensor_{10}$ acts independently within the cycle starting with the action

gather_data. Similarly for $Sensor_{20}$. Thus we can see that R is also a guarding and returning resource with respect to the model component ($Sensor_{10} \parallel Sensor_{20}$).

Considering the sensors in isolation we can deduce that their equilibrium probability distributions are as follows:

$$\begin{aligned}
\pi_1(S_{110}, S_{120}) &= r_2 r_3 r_4 (r_2 + r_4)/G_1 \\
\pi_1(S_{111}, S_{121}) &= r_1 r_2 r_3 r_4 / G_1 \\
\pi_1(S_{112}, S_{121}) &= r_1 r_2^2 r_3 / G_1 \\
\pi_1(S_{111}, S_{122}) &= r_1 r_3 r_4^2 / G_1 \\
\pi_1(S_{112}, S_{122}) &= r_1 r_2 r_4 (r_2 + r_4)/G_1 \\
\pi_1(S_{113}, S_{120}) &= r_3 r_4 r_5 (r_2 + r_4)/G_1
\end{aligned}$$

where $G_1 = (r_2 r_4 + r_4^2)(r_1 r_2 + r_1 r_3 + r_2 r_3 + r_3 r_5) + r_1 r_2^2 r_3$.

$$\begin{aligned}
\pi_2(S_{210}, S_{220}) &= r_3 r_6 r_7 (r_6 + r_7)/G_2 \\
\pi_2(S_{211}, S_{221}) &= r_1 r_3 r_6 r_7 / G_2 \\
\pi_2(S_{212}, S_{221}) &= r_1 r_3 r_6^2 / G_2 \\
\pi_2(S_{211}, S_{222}) &= r_1 r_3 r_7^2 / G_2 \\
\pi_2(S_{212}, S_{222}) &= r_1 r_6 r_7 (r_6 + r_7)/G_2
\end{aligned}$$

where $G_2 = (r_6 r_7 + r_7^2)(r_1 r_3 + r_1 r_6 + r_3 r_6) + r_1 r_3 r_6^2$.

For the complete system, from Theorem 16 we can deduce that the equilibrium probability of any state $(S_{11i}, S_{12j}, S_{21k}, S_{22l}, R')$ is:

$$\pi(S_{11i}, S_{12j}, S_{21k}, S_{22l}, R') = B\pi_1(S_{11i}, S_{12j}) \times \pi_2(S_{21k}, S_{22l})$$

Here we have solved a six state model and a five state model instead of a 14 state model. However in general the ability to decompose the solution of a model in this way may mean the difference between the model being tractable and intractable.

6 Conclusions and Further Work

We have demonstrated a class of PEPA models which satisfy Boucherie's conditions and so can be decomposed into submodels which are then solved separately without loss of accuracy. Unlike the Petri net models presented in [3], the PEPA models which exhibit product form can be detected automatically. Unlike the Markov processes presented in [3] all elements of the system, including the resources, are explicitly and formally represented in the model. The advantage of using MPA for this work is the formality of the approach and

the subsequent automation of the technique within a modelling tool, such as the PEPA Workbench [8]. In order to tackle models of the size and complexity needed for the performance analysis of the next generation of computer systems, efficient techniques for the construction, manipulation and solution of large Markov processes will be required. PEPA provides a formal, structured, yet expressive, high-level language for specifying such processes.

Furthermore, as we have seen in Section 5.2 the MPA context provides a framework for comparing models which satisfy different product form criteria. The taxi model presented in that section can clearly be seen to satisfy Sereno's criteria for product form as well as those presented in this paper. In future work we hope to use the MPA context to investigate fully the relationship between the different classes of product form model which have been identified.

Several issues remain to be explored with respect to the class of models identified in this paper. Perhaps the most important one is to establish an efficient method for calculating the normalising constant. There are several published algorithms which could be employed in this regard, including one recently presented by Coury and Harrison [6]. We also intend to extend the class of models considered by allowing multiple instances of resource components.

Although developed here in the context of PEPA, our characterisation of components competing over resources is equally applicable to the other MPA languages, if we restrict to the strictly Markovian cases. The main difference between the languages lies in the rate of shared activities but this detail of model behaviour is unimportant within the class of models we consider. However in EMPA, and some versions of MTIPP, immediate actions are included; the effect of allowing such actions to form the basis of cooperation between the resource and the quasi-independent components of the model would need to be investigated.

The class of models presented here is quite limited. However a much larger class of models *almost* fit these criteria. Through tool support within the PEPA Workbench we aim, in future, to be able to identify this wider class of models and use the decomposition technique presented here to provide reliable approximate solutions. This work forms part of a wider goal of implementing a wide range of decomposition techniques for product form and almost product form models. An account of the development of the PEPA Workbench in this regard can be found in [5].

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (COMPA project, G/L10215). It originated from, and benefited greatly from, discussions with Matteo Sereno of University of Torino, made possible by funding from The British Council, CNR (ROM/889/94/9) and the QMIPS BRA project. We would also like to thank Graham Clark, Stephen Gilmore and the anonymous referees for helpful comments and suggestions during the preparation of this paper.

References

[1] M. Bernardo and R. Gorrieri, A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time, *Theoretical Computer Science* **202**(1-2) (1998) 1–54.

[2] M. Bhabuta, P.G. Harrison and K. Kanani, Detecting reversibility in Markovian Process Algebra, in: M. Merabti, M. Carew and F. Ball eds., *Performance Engineering of Computer and Telecommunications Systems*, Springer, 1995.

[3] R. Boucherie, A Characterisation of Independence for Competing Markov Chains with Applications to Stochastic Petri nets, *IEEE Transactions on Software Engineering* **20**(7) (1994) 536–544.

[4] P. Buchholz, Hierarchical Markovian Models - Symmetries and Reduction, in: R. Pooley and J. Hillston, eds., *Computer Performance Evaluation: Modelling Techniques and Tools* (Edinburgh University Press, 1993) 305–319.

[5] G. Clark, S. Gilmore, J. Hillston and N. Thomas, Experience with the PEPA Performance Modelling Tools, in: R. Pooley and N. Thomas, eds., *Proc. of the 14th UK Performance Engineering Workshop* (Edinburgh, July 1998).

[6] S. Coury and P. Harrison, On the Asymptotic Behaviour of Closed Multiclass Queueing Networks, in: R. Zobel and D. Moeller, eds., *Proc. of 12th European Simulation Multiconference* (Manchester, June 1998).

[7] S. Donatelli, Superposed Generalised Stochastic Petri Nets: Definition and Efficient Solution, in: M. Silva, ed., *Proc. of 15th Int. Conf. on Application and Theory of Petri Nets* (1994).

[8] S. Gilmore and J. Hillston, The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling, in: G. Haring and G. Kotsis, eds., *Proc. of 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation* (Lecture Notes in Computer Science, vol. 794, Springer, 1994) 353–368.

[9] S. Gilmore, J. Hillston, R. Holton, and M. Rettelbach. Specifications in Stochastic Process Algebra for a Robot Control Problem, *International Journal of Production Research* **34**(4) (1996) 1065-1080.

[10] N. Götz, U. Herzog and M. Rettelbach, TIPP - a language for timed processes and performance evaluation, in: J. Hillston and F. Moller, eds., *Proc. of 1st Workshop on Process Algebra and Performance Modelling* (University of Edinburgh, May 1993).

[11] P. Harrison and J. Hillston, Exploiting quasi-reversible structures to find product form solutions in Markovian process algebra models, *The Computer Journal* **38**(7) (1995) 510–520.

[12] W. Henderson and P.G. Taylor, Embedded Processes in Stochastic Petri Nets, *IEEE Transactions on Software Engineering* **17**(2) (1991) 108–116.

[13] J. Hillston, The Nature of Synchronisation, in: U. Herzog and M. Rettelbach, eds., *Proc. of 2nd Workshop on Process Algebra and Performance Modelling* (Regensburg, Germany, July 1994).

[14] J. Hillston, *A Compositional Approach to Performance Modelling* (Cambridge University Press, 1996).

[15] J. Hillston, Compositional Markovian Modelling Using a Process Algebra, in: W. Stewart ed., *Computations With Markov Chains* (Kluwer, 1995).

[16] J. Hillston, A Class of PEPA Models Exhibiting Product Form Solution over Submodels, (Technical Report LFCS-98-382, Department of Computer Science, University of Edinburgh, 1998).

[17] J. Hillston, Exploiting Structure in Solution: Decomposing Composed Models, in *Proc. of 6th Int. Workshop on Process Algebra and Performance Modelling* (Nice, France, September 1998).

[18] J. Hillston and N. Thomas, A Syntactic Analysis of Reversible PEPA Processes, in *Proc. of 6th Int. Workshop on Process Algebra and Performance Modelling* (Nice, France, September 1998).

[19] D.R.W. Holton, A PEPA Specification of an Industrial Production Cell, *The Computer Journal* **38**(7) (1995) 542–551.

[20] A. Lazar and T. Robertazzi, Markovian Petri Net Protocols with Product Form Solution, *Performance Evaluation* **12**(1) (1991) 67–77.

[21] V. Mertsiotakis, Approximate Analysis Methods for Stochastic Process Algebras. (PhD thesis, Universität Erlangen-Nürnberg, Martensstraße 3, 91058 Erlangen, September 1998).

[22] R. Milner, *Communication and Concurrency* (Prentice-Hall, 1989).

[23] M. Ribaudo, On the Aggregation Techniques in Stochastic Petri Nets and Stochastic Process Algebras, *The Computer Journal* **38**(7) (1995) 600–611.

[24] W. Sanders and J. Meyer, Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications* **9**(1) (1991) 25–36.

[25] M. Sereno, Towards a product form solution for stochastic process algebras, *The Computer Journal* **38**(7) (1995) 622–632.

A Structured Operational Semantics for PEPA

The semantic rules, in the structured operational style of Plotkin, are presented here without comment; the interested reader is referred to [14] for more details. The rules are read as follows: if the transition(s) above the inference line can be inferred, then we can infer the transition below the line.

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'} \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \ (\alpha \notin L) \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \ (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \ (\alpha \in L), \qquad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \ (\alpha \notin L) \qquad \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \ (\alpha \in L)$$

Constant

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

Fig. A.1. Operational Semantics of PEPA

In the third rule for cooperation we make use of the notion of *apparent rate*, to define the rate of a shared activity. The apparent rate of an action type

represents the totally capacity of a component to carry out activities of that type in its current state.

Definition 15 (Apparent Rate) *The* apparent rate *of action of type α in a component P, denoted $r_\alpha(P)$, is the sum of the rates of all activities of type α in $\mathcal{A}ct(P)$.*

In cooperation the apparent rate of the shared activity will be the minimum of the apparent rates of the components involved, where $\min(\top, r) = r$ for all $r \in \mathbb{R}^+$. Thus the rules embody an assumption that shared activities proceed at the rate of the slower of the two participating components. This is based on a notion that in general both components contribute some work to the shared activity. For a discussion of alternative assumptions see [13]. In the case of a passive action it is assumed that the corresponding component does not contribute at all to the work required to complete the shared activity.