Extending Quasi-Separability

Nigel Thomas *

Abstract

The advantages of compositionality and flexibility of stochastic process algebras has led to them being used to model increasingly larger and more complex systems. Models of such systems are inherently difficult to solve due principally to the size of their underlying state space. Therefore, tools which support the evaluation of models expressed using stochastic process algebra must be able to utilise the full range of decomposition and solution techniques available. In this paper a class of models is considered which do not give rise to a product form solution but can nevertheless be decomposed into their components without loss of generality. The characterisation developed here generalises earlier published results, allowing a much wider range of models to be decomposed in this way.

1 Introduction

In the study of stochastic process algebra we wish to consider not only how systems are to be specified, but also how complex systems can be simplified and solved efficiently. One part of this research has been the study of stochastic process algebra models that give rise to product form solutions (see [4] for a review of these results). However, the class of performance models exhibiting product form is limited, moreover it is not always necessary to derive a full product form decomposition to find performance measures of interest. Some stochastic process algebra work on product form has been based on well known cases of product form solution in queueing network models. In this paper an alternative method of model decomposition is considered that can be found in the queueing network literature, *quasi-separability*.

Quasi-separability was developed in the study of systems which suffer breakdowns [6, 8], more recently the approach has been applied by Thomas and Gilmore [7] to models specified using the Markovian process algebra PEPA. This characterisation is based on the decomposition of queueing models into submodels each involving one individual queue, however, by using PEPA a much more general type of component can be considered. More specifically, the characterisation of a quasi-separable Markov process in [7] involves components whose state is represented by a pair of integers, whereas in general it is possible to consider a much more complex class of components. In general there is no bound to the number of variables which describe the state of a component, the only restriction placed is on the nature of the interaction between components. In this paper such a class of models is considered.

*Dept. of Computer Science, University of Durham

This approach is illustrated using a model of parallel fault-protected gateways to a network. Messages are generated by users requesting access to the network and are routed through one of the parallel gateways. The gateways are subject to failures that interrupt access to the network for random periods of time. Messages may be directed away from gateways that are known to be faulty, but messages already awaiting access at a faulty gateway are held until repair is complete. No messages are lost except when directed to a queue which is already full. In general the gateways are subject to failures and repairs of different rates. In addition messages are processed at different rates at different gateways, but the service they receive is functionally equivalent. This system is modelled as a system of M/M/1/N queues in parallel.

This example is similar to the existing queueing literature involving quasi-separability, therefore a second example is presented to further illustrate the approach. This example concerns a system of distributed sensors operating in a changing environment. In this model the communication between sensors and a central server is unreliable. This is a common factor in remotely sited sensing equipment when many factors may inhibit communication, for example weather conditions, electronic interference, geological disturbance and of course the reliability of equipment left in the field. This model is shown to satisfy the criteria for quasi-separability so that the performance of each sensor may be studied in semi-isolation.

Both the examples illustrate the point that quasi-separability is applicable to models which fall outside criteria for a product form solution. As such it is clear that the characterisation presented here, while limited in its applicability, complements existing product form results for stochastic process algebra.

In the following sections the concept of quasi-separability is introduced, a brief overview of the Markovian process algebra PEPA is given, a characterisation of quasi-separable PEPA models is defined and the approach is illustrated through two examples.

2 Quasi-Separability

The concept of quasi-separability is closely linked to that of product form, in particular product form over resources discussed by Boucherie [1] and applied to stochastic process algebra by Hillston and Thomas [5]. A decomposition based on quasi-separability allows expressions to be derived for marginal distributions just as with a product form solution, however unlike product form these marginal distributions cannot, in general, be combined to form the joint distribution for the whole model. Despite the lack of a solution for the joint distribution, many performance measures of interest can still be derived exactly. In particular it is possible to obtain performance measures in the form of long run averages, such as average response times in a queueing network.

2.1 Properties of quasi-separable Markov processes

Informally a system which is amenable to a quasi-separable solution can be considered in the following way. The entire system operates within a single environment, which may be made up of several sub-environments. Several components operate within this environment such that their behaviour is affected by the state of the environment. The state of each component does not alter the state transitions of either the environment or the other components. The behaviour of such components can clearly be studied in isolation from the other components as long as the state of the environment is considered also. The restriction on the behaviour of the components is unnecessarily strong. We can also consider models where the state space of the components can be separated into that part which does have an impact on state transitions in the environment or other components and that part which has no external influence, not even on the other part of that component. Such a separation requires that the part of a component that influences the state of the environment is considered to be part of the environment for the purposes of model decomposition.

Models such as these have appeared in the literature of the study of queueing systems with breakdowns and rerouting of jobs [6, 8]. In such models the environment is generally made up of the operational state of servers in the system. For instance each server might be either working or broken, so for a system of N servers the environment has 2^N states. The routing of jobs to queues is dependent on the operational state of the system i.e. the state of the environment. Such models can generally be decomposed into single queue systems with Markov-modulated arrivals and breakdowns. This type of model is conceptually quite simple, there are only two aspects to the state of the components. A characterisation based on this simple view of quasi-separability was presented in [7] in the context of the Markovian process algebra PEPA. We now wish to extend this characterisation to the more general form discussed informally above.

Consider an irreducible Markov process, $X(t)$, which consists of N separate components. The state of each component i can be described by a set of K_i separate variables. Denote by $\mathcal{V}_i$ the set of K_i variables which describe the state of component i. If it is possible to analyse the behaviour of each component, i, of the system exactly by only considering those variables that describe it, i.e. $\mathcal{V}_i$, then the system is said to be *separable*. In this case all the components are statistically independent and a product form solution exists.

For the system to be *quasi-separable* it is necessary only that it is possible to analyse the behaviour of each component, i, of the system exactly by only considering those variables that describe it, $\mathcal{V}_i$, and a subset of the variables from all the other components. Thus the elements of $\mathcal{V}_i$ can be classified into the subsets of either system state variables, $\mathcal{S}_i$ or component state variables $\mathcal{C}_i$, such that:

- the state of $c(t) \in \mathcal{C}_i$ changes at a rate which is independent of the state of any variable $v(t) \in \mathcal{C}_j$, $\forall\, j$ s.t. $j \neq i$.
- the state of $s(t) \in \mathcal{S}_i$ changes at a rate which is independent of the state of any variable $v(t) \in \mathcal{C}_j$, $1 \leq j \leq N$.

If $\mathcal{C}_i \neq \emptyset$, $\forall\, i$, the system can be decomposed into N submodels such that the submodel of the system with respect to the behaviour of component i specifies the changes in the system state variables $\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i$ and the component state variables $\mathcal{C}_i$. In general the analysis of these submodels gives rise to expressions for their steady-state marginal probabilities if the submodels have stationary distributions with state spaces which are infinite in at most one dimension. As stated above, these marginal probabilities do not, in general, give rise to expressions for the joint probability of the whole system, i.e. no product form solution exists. For quasi-separability to be useful the state space of the submodels should be significantly smaller than the state space of the entire model.

3 PEPA

A formal presentation of PEPA is given in [3], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the *type* of the activity, α, and the *rate* of the associated negative exponential distribution, r. This rate may be any positive real number, or given as *unspecified* using the symbol $\top$. The syntax for describing components is given as:

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

The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P. The component $P + Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete. The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ. Concurrent components can be synchronised, $P \underset{L}{\bowtie} Q$, such that activities in the *cooperation set* L involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to that activities of that type. The parallel combinator $||$ is used as shorthand to denote synchronisation with no shared activities, i.e. $P||Q \equiv P \underset{\emptyset}{\bowtie} Q$. $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P.

In the following sections two PEPA expressions will be referred to as being *isomorphic*. To formally define this equivalence relation it is necessary to define the semantics of the language. Since these definitions are presented in detail elsewhere [2, 3], they are omitted here. Informally, it can be said that, two isomorphic PEPA expressions give rise to Markov chains which are equivalent such that for every state in each Markov chain there is a corresponding state in the other with the same one-step transition rates to states which are similarly equivalent. The *derivative* of a component is also referred to, again this is defined formally in [3]. Informally a derivative is the "state" of a component defining its current behaviour. For example, if $P \stackrel{def}{=} (\alpha, r).Q$ then Q is a derivative of P and if $Q \stackrel{def}{=} (\beta, r_2).R$ then R is a derivative of both Q and P, and so on.

In the following sections only models which have a steady state solution are considered, necessary conditions for which are given in [3].

4 Quasi-separable PEPA components

In this section a characterisation of quasi-separability in PEPA is derived. This characterisation relies on the fact that the model specification in PEPA exactly mirrors the definition of a quasi-separable Markov process given in Section 2. However, in reality there are a great many possible specifications for any given model and the exact criteria for quasi-separability are not, as yet, fully defined. It is therefore understandable that this characterisation captures a very limited class of models. As this and other similar decomposition techniques are studied further this characterisation and the understanding of the criteria for decomposition should greatly improve.

A system of N components expressed in PEPA is now considered. The components are modelled as PEPA components, $C_1, \ldots, C_N$ say. The interactions of these components is coordinated by another component S, which is referred to as the scheduler. These components are related to the underlying Markov process of the form discussed in Section 2 such that, S represents all the system state variables $\mathcal{S}$ and C_i represents *at least* all the component state variables $\mathcal{C}_i$, $i = 1, \ldots, N$. That is, the C_i's represent those parts of the system that are considered in isolation and S represents those parts which are not.

No actions are synchronised between the components $C_1, \ldots, C_N$. All actions enabled in C_i are either synchronised between it and S, contained in the set L, or are unique to C_i. Only actions which can be internalised are enabled in the scheduler, S. Any actions in the cooperation set L either do not change S or, if they do, they cannot be blocked by C_i. Thus the system can be described in the following way;

$$(C_1 \,||\, \cdots \,||\, C_N) \underset{L}{\bowtie} S \tag{4.1}$$

The set of synchronised actions L can be rewritten as N subsets L_i, $i = 1, \ldots, N$. The subset L_i contains only those actions that are synchronised between S and C_i i.e. those actions in L and enabled in C_i (and not hidden). Denote by Z_i the set of all actions that are synchronised between S and each C_j such that $j \neq i$, i.e.

$$Z_i = \bigcup_{\forall\, j \neq i} L_j$$

Clearly, $Z_i \subset L$ and $L = (L_i \cup Z_i)$, $i = 1, \ldots, N$. If the actions in subsets Z_i are such that their rates are not influenced by C_i, i.e.

- $L_i \cap Z_i = \emptyset$, or
- l is enabled in every derivative of C_i, $1 \leq i \leq N$, with rate $\top$, $\forall\, l \in (L_i \cap Z_i)$.

The complete model specification can then be rewritten in the following way;

$$(C_1 \underset{L_1}{\bowtie} S) \underset{Z_1}{\bowtie} (C_2 \,||\, \cdots \,||\, C_N) \tag{4.2}$$

such that (4.1) and (4.2) are *isomorphic*. Therefore, the subsystem, $(C_1 \underset{L_1}{\bowtie} S)$ can now be isolated from the system without altering the individual behaviour of C_1 in any way, subject to the conditions for L_1 and Z_1 given above.

The same approach can be applied for every C_i, L_i and Z_i to give expressions for all N subsystems. A system that can be treated in such a way satisfies the conditions for *quasi-separability* presented in Section 2.

In the characterisation presented above the separation of system state and component variables has already been made. An alternative characterisation is possible whereby the model is expressed as the cooperation of N components.

$$V_1 \underset{Y_1}{\bowtie} V_2 \underset{Y_2}{\bowtie} \cdots \underset{Y_{N-1}}{\bowtie} V_N \;\; where, \; V_i \stackrel{def}{=} S_i \underset{W_i}{\bowtie} C_i \;\; , \;\; 1 \leq i \leq N$$

such that, S_i represents all the system state variables $\mathcal{S}_i$ and C_i represents *at least* all the component state variables $\mathcal{C}_i$, $i = 1, \ldots, N$. There are two possible cases for the synchronisation set W_i for which quasi-separability may hold;

- no action of type $w \in W_i$ causes a change in the behaviour of S_i
- any action of type $w \in W_i$ which causes a change in the behaviour of S_i is enabled in every derivative of C_i with rate $\top$.

An additional pathological case exists whereby an action of type $w \in W_i$ is blocked in every possible combination of S_i and C_i, i.e. when actions of type $w \in W_i$ have no effect within the model. Thus it is assured that the behaviour of S_i is independent of the behaviour of C_i, but the converse is not generally true. Furthermore it is necessary to ensure that the behaviour of S_i is independent of the state of any C_j, $1 \leq j \leq N$. Therefore the cooperation set, Y_i, between components is similarly restricted such that any action of type $y \in Y_i$ which is enabled in C_j, $1 \leq j \leq N$ is enabled in every derivative of C_j with rate $\top$. Hence it is possible to form an expression for S which represents all the system state variables $\mathcal{S}$.

$$S \stackrel{def}{=} S_1 \underset{Y_1}{\bowtie} S_2 \underset{Y_2}{\bowtie} \cdots \underset{Y_{N-1}}{\bowtie} S_N$$

Furthermore, since any shared actions which are enabled in C_i, $1 \leq i \leq N$ are enabled in every derivative of C_i with rate $\top$, there are no shared actions in C_j, $1 \leq j \leq N$, which need the cooperation of C_i. Hence the model can be rewritten as,

$$(C_1 \| \cdots \| C_N) \underset{W \cup Y'}{\bowtie} S$$

where $W = \bigcup_{i=1}^{N} W_i$ and $Y' \subset \bigcup_{i=1}^{N} Y_i$ such that an action of type $y \in Y'$ is enabled in at least one of the C_i's $(1 \leq i \leq N)$. This expression can now be rewritten in the same way as for the previous case to give,

$$C'_j \stackrel{def}{=} (C_1 \| \cdots \| C_{j-1} \| C_{j+1} \| \cdots \| C_N)$$
$$(C_j \underset{L_j}{\bowtie} S) \underset{Z_j}{\bowtie} C'_j$$

where,

$$\begin{aligned}
L_j &= W_j \cup Y'_j \\
Z_j &= (W_1 \cup \cdots \cup W_{j-1} \cup W_{j+1} \cup \cdots \cup W_N) \cup X_j \\
X_j &= Y'_1 \cup \cdots \cup Y'_{j-1} \cup Y'_{j+1} \cup \cdots \cup Y'_N \\
Y'_j &\subset \bigcup_{i=j-1}^{N} (Y_i \cap Y') \text{ such that an action of type } y \in Y'_j \text{ is enabled} \\
&\quad\ \text{(and not hidden) in } C_j \ (Y_0 = \emptyset).
\end{aligned}$$

Hence the set W_j denotes the actions shared between C_j and S_j, Z_j denotes the actions shared between the C_i's $(i \neq j)$ and S, and Y'_j denotes the actions shared between C_j and S_i $(\forall\, i \neq j)$.

Clearly the characterisations presented here are in an idealised form and requires further development for much more general models to be considered. However the characterisation of quasi-separable PEPA components given here does not severely restrict the structure of the components in any way other than through the actions which are shared between them.

5 Examples

In this section the application of quasi-separability is illustrated using two examples. The models presented here are shown to be of the type defined in Section 4 and can therefore be decomposed into a set of submodels which can be solved numerically much more easily.

5.1 Routing between unreliable parallel gateways

Consider a system where jobs from a common incoming stream may be directed to one of N alternative gateways, each of which consists of a single server and a bounded queue. The service, breakdown and repair processes at the different nodes are independent of each other and have different parameters, in general. The consequences of a breakdown at a server are not too catastrophic: service stops and the existing jobs remain in place; new arrivals during the subsequent repair period may be re-directed to other queues. There are no job losses except when jobs are sent to a queue which is already full.

Jobs arrive into the system in a Poisson stream with rate λ. There are N servers, each with an associated bounded queue, to which incoming jobs may be directed. Server k goes through alternating independent operative and inoperative periods, distributed negative exponentially with means $1/\xi_k$ and $1/\eta_k$, respectively. While it is operative, the jobs in its queue receive negative exponentially distributed services with mean $1/\mu_k$, and depart upon completion. When a server becomes inoperative (breaks down), the corresponding queue, including the job in service (if any), remains in place. Services that are interrupted in this way eventually resume from the point of interruption.

The *system configuration* at any moment is specified by the subset, σ, of servers that are currently operative (that subset may be empty, or it may be the set of all servers): $\sigma \subset \Omega_N$, where $\Omega_N = \{1, 2, \ldots, N\}$. There are of course 2^N possible system configurations. If, at the time of arrival, a new job finds the system in configuration σ, then it is directed to node k with probability $q_k(\sigma)$. These decisions are independent of each other, of past history and of the sizes of the various queues. Thus, a routing policy is defined by specifying 2^N vectors,

$$\mathbf{q}(\sigma) = [q_1(\sigma), q_2(\sigma), \ldots, q_N(\sigma)] \ , \ \ \sigma \subset \Omega_N \ , \tag{5.1}$$

where $q_k(\sigma)$ is the probability that a job is directed to the kth queue, such that for every σ, $\sum_{k=1}^{N} q_k(\sigma) = 1$.

Intuitively, it seems better not to send jobs to nodes where the server is inoperative, unless that is unavoidable. This suggests the following strategy. If the subset of operative servers in the current system configuration is σ, and that subset is non-empty, send jobs to node k only if $k \in \sigma$, with probability proportional to q_k :

$$q_k(\sigma) = \frac{q_k}{\sum_{\ell \in \sigma} q_\ell} \ , \ \ k \in \sigma \ ,$$

$q_k(\sigma) = 0$, otherwise. If σ is empty (i.e. all servers are broken), send jobs to node k with probability q_k ($k = 1, 2, \ldots, N$). This last point preserves the system property of being *fault protected*, i.e. the effect of a failure is not catastrophic. A specification of this problem in the Markovian process algebra PEPA is given in Figure 1, for the case where there are two queues in parallel.

$$
\begin{aligned}
Queue1_0 &\stackrel{def}{=} (arr1, \top).Queue1_1 \\
Queue1_j &\stackrel{def}{=} (arr1, \top).Queue1_{j+1} + (serv1, \top).Queue1_{j-1}\ ,\ 1 \le j \le n-1 \\
Queue1_n &\stackrel{def}{=} (arr1, \top).Queue1_n + (serv1, \top).Queue1_{n-1} \\[1em]
Queue2_0 &\stackrel{def}{=} (arr2, \top).Queue2_1 \\
Queue2_j &\stackrel{def}{=} (arr2, \top).Queue2_{j+1} + (serv2, \top).Queue2_{j-1}\ ,\ 1 \le j \le n-1 \\
Queue2_n &\stackrel{def}{=} (arr2, \top).Queue2_n + (serv2, \top).Queue2_{n-1} \\[1em]
Sigma_0 &\stackrel{def}{=} (arr1, \lambda q_1).Sigma_0 + (arr2, \lambda q_2).Sigma_0 \\
&\quad +(repair1, \eta_1).Sigma_1 + \ (repair2, \eta_2).Sigma_2 \\
Sigma_1 &\stackrel{def}{=} (serv1, \mu_1).Sigma_1 + (arr1, \lambda).Sigma_1 + (fail1, \xi_1).Sigma_0 \\
&\quad + (repair2, \eta_2).Sigma_3 \\
Sigma_2 &\stackrel{def}{=} (serv2, \mu_2).Sigma_2 + (arr2, \lambda).Sigma_2 + (fail2, \xi_2).Sigma_0 \\
&\quad + (repair1, \eta_1).Sigma_3 \\
Sigma_3 &\stackrel{def}{=} (arr1, \lambda q_1).Sigma_3 + (arr2, \lambda q_2).Sigma_3 + (serv1, \mu_1).Sigma_3 \\
&\quad + (serv2, \mu_2).Sigma_3 + (fail1, \xi_1).Sigma_2 + (fail2, \xi_2).Sigma_1
\end{aligned}
$$

$$
(Queue1_0 \,||\, Queue2_0) \underset{\{arr1,\, arr2,\, serv1,\, serv2\}}{\bowtie} Sigma_3
$$

Figure 1: A PEPA model of two $M/M/1$ queues in parallel with state dependent routing

As with most languages, there are many ways to describe this model in PEPA. However, the aim here is not only to produce a concise specification, but also one that captures a specific view of this model and facilitates simplification and solution in a clear manner. A specification is proposed consisting of passive buffers controlled by a scheduler which represents the operational state of the system, σ. The index of the scheduler is an integer representation of the binary system operational state, σ, where broken is 0 and not broken is 1, e.g. if only server i is working the operational (scheduler) state is 2^i and so on. Expressions for the scheduler can readily be generated recursively, as can expressions for additional queues. Therefore the description of the system given here can easily be expanded. An alternative approach to describing such a model might be to consider each of the servers as separate processes, rather than collectively in the scheduler. Such an approach would conceivably give rise to fewer terms in the specification when the number of queues is large.

In all cases the number of jobs in each queue is dependent on the actions of the scheduler, therefore they are not separable. However, the queues themselves can be considered as independent of each other and still maintain the same overall behaviour, i.e. this model exhibits quasi-separability. By considering the queues in this way it is possible to find the marginal queue size probabilities, which can then be used to find most performance measures of interest.

The actions $arr1$ and $serv1$ apply only to queue 1 and the actions $arr2$ and $serv2$ apply only to queue 2. Hence the expressions,

$$(Queue1_0 \,||\, Queue2_0) \underset{\substack{\{arr1, arr2,\\ serv1, serv2\}}}{\bowtie} Sigma_3$$

$$(Queue1_0 \underset{\{arr1, serv1\}}{\bowtie} Sigma_3) \underset{\{arr2, serv2\}}{\bowtie} Queue2_0$$

and,

$$(Queue2_0 \underset{\{arr2, serv2\}}{\bowtie} Sigma_3) \underset{\{arr1, serv1\}}{\bowtie} Queue1_0$$

are isomorphic. Thus, evaluating the first part of each of these expressions,

$$Queue(k)_0 \underset{\{arr(k), serv(k)\}}{\bowtie} Sigma_3$$

will give an exact result for the marginal queue size probabilities for queue k. Clearly if models have been specified with passive actions in the scheduler then the scheduler expressions will need to be rewritten before evaluating the synchronised process.

It is important to note that this approach does not give rise to expressions for the joint queue size probabilities since, as stated, this system is not separable. However the average number of jobs in the system is given by the sum of the average number of jobs in each queue which can be found from the marginal queue size probabilities. Other performance measures of interest, such as the average rate of job loss and average response time, can also be derived from these marginal probabilities. It is clear, therefore, that this approach does not give a product form solution, but is very useful nevertheless.

Clearly it would be possible, in this case, to obtain a numerical solution for the entire model. However, if the number of states in each queue was large then such a solution would be extremely costly. Furthermore if the size of the queues was unbounded, i.e. $n \longrightarrow \infty$ then an exact numerical solution to this model would not generally be tractable. The state space of the underlying CTMC for the entire systems has $(2n+2)^N$ states whereas applying quasi-separability gives rise to N submodels each with an underlying CTMC of $2^N(n+1)$ states.

5.2 Distributed data collection system

Now consider an example of a system where communication between remotely situated sensing devices and a centrally located server is unreliable. These sensors may become overloaded if the data which they collect cannot be downloaded. In such a situation they will not be able to collect data and will need to be reset by the central server for normal operation to resume. There are many reasons for interrupting communication between such distributed units, these include, weather conditions, electrical interference, geological disruptions and equipment failure.

The structure of this model is as follows. In normal operation sensing devices are continually collecting data from their local environment. Each sensor is periodically polled in turn to download data to a central server, whereupon the sensor is reset. In addition there are randomly occurring interruptions to the communication between the central server and the sensors. These interruptions are local, affecting communication with just a single sensor. These interruptions do not affect the ability of the sensors to collect data. However the sensors have limited ability to store data, therefore if the sensor is not contacted by the central server for some time it may become overloaded. In such situations it will need to be reset by the central server for normal data collection to resume.

There are several possible polling policies the central server could adopt in the event of a communication interruption. For a local interruption these include:

- simply miss the sensor out of the normal polling sequence until it is available;
- contact the affected sensor immediately it becomes available.

Clearly there are several further possibilities and the situation is greatly complicated if there are simultaneous local and global interruptions or if the central server is unaware of the nature of the interruption. In a performance study of such a system several policies would be compared to deduce the optimum performance over a given set of criteria. Since we are only interested here in how such a model can be decomposed only the situation where in the event of a local interruption no remedial action will be taken by the central server will be considered. Such a model is illustrated in Figure 2.

$$
\begin{array}{rcl}
Sensor_{j,0} & \stackrel{def}{=} & (collect, \gamma_j).Sensor_{j,1} + (downld_j, \top).Sensor_{j,0} \\
Sensor_{j,k} & \stackrel{def}{=} & (collect, \gamma_j).Sensor_{j,k+1} + (downld_j, \top).Sensor_{j,0} \\
 & & 1 \leq k \leq M-1 \\
Sensor_{j,M} & \stackrel{def}{=} & (collect, \gamma_j).Overload_j + (downld_j, \top).Sensor_{j,0} \\
Overload_j & \stackrel{def}{=} & (downld_j, \top).Sensor_{j,0} \\
 & & \\
NoIntrpt_j & \stackrel{def}{=} & (downld_j, \top).NoInterrupt_j + (breakdn, \xi_j).Interrupt_j \\
Interrupt_j & \stackrel{def}{=} & (block_j, \top).Interrupt_j + (end, \eta_j).NoIntrpt_j \\
 & & \\
Interrupt & \stackrel{def}{=} & (Interrupt_1 \,\|\, \cdots \,\|\, Interrupt_N) \\
 & & \\
Server_j & \stackrel{def}{=} & (downld_j, \mu).Server_{j+1} + (block_j, \mu).Server_{j+1} \\
 & & 1 \leq j \leq N \\
Server_{N+1} & \stackrel{def}{=} & (wait, \alpha).Server_1
\end{array}
$$

$$
(Sensor_{1,0} \,\|\, \cdots \,\|\, Sensor_{N,0}) \underset{\{downld_1,\ldots,downld_N\}}{\bowtie} (Server_1 \underset{\{downld_1,\ldots,downld_N,\, block_1,\ldots,block_N\}}{\bowtie} Interrupt)
$$

Figure 2: A PEPA model of N distributed sensors with a central server and interrupted communication

The structure of the model in this example is slightly more complicated than in the previous case. In the previous example there were just two categories of components, the queues and the operational state. In this example there are three categories of components, the sensors, the central sever and the interruptions in communication between the server and the sensors. Also the action $download_j$ may cause a change in the behaviour of $Server_j$, where $j = 1, \ldots, N$, and is enabled passively in every derivative of $Sensor_{j,k}$, $k = 0, \ldots, M$.

The behaviour of each sensor is dependent on the behaviour of the server and the possibility of any interruption between sensor and server. Furthermore it is apparent that the presence of any interruption will affect what actions are carried out by the server, hence each submodel will need to contain the jth sensor, all the interrupts and the server.

Hence, following the same argument as previously, we can extract the submodels,

$$Sensor_{j,0} \underset{\{downld_j\}}{\bowtie} (Server_1 \underset{\substack{\{downld_1,\ldots,downld_N,\\ block_1,\ldots,block_N\}}}{\bowtie} Interrupt) \tag{5.2}$$

From these expressions performance measures can be calculated, such as the average probability of overload and the average amount of valid data transferred to the central server without the need to explore the entire state space of the full model. As in the previous example it is theoretically possible to derive an exact solution to the full model. However, the full model, as illustrated in Figure 2, has $2^N(M+2)^N(N+1)$ states in the underlying CTMC, whereas each submodel, given by (5.2), has $2^N(M+2)(N+1)$ states. Clearly therefore there is a considerable saving in the state space to be made and this saving is dependent on the number of sensors in the system and the amount of storage space each sensor has available.

Such a decomposition is possible regardless of the rates of the actions in $Server_j$. However, it is interesting to note that there is an equivalent behaviour between the actions $downld_j$ and $block_j$ in $Server_j$ in Figure 2. The server is "unaware" of any interruptions and so the rates of its actions $download_j$ and $block_j$ are identical, as is the subsequent behaviour of $Server_j$. Thus it is clear that the presence of any interruptions does not affect the outward behaviour of $Server_j$, only the type of action which may occur. Hence it is possible to consider submodels with $2(M+2)(N+1)$ states of the form:

$$Sensor_{j,0} \underset{\{downld_j\}}{\bowtie} (Server_1 \underset{\substack{\{downld_j,\\ block_j\}}}{\bowtie} Interrupt_j)$$

6 Concluding Remarks

In this paper it has been shown how complex models expressed in PEPA can be greatly simplified by applying quasi-separability. Decompositions of this kind are extremely useful when tackling models with large state spaces, especially when the state space grows exponentially with the addition of further components. Quasi-separability can be applied to a wide range of models to derive numerical results very efficiently. While it does not generally give rise to expressions for joint probability distributions it does provide exact results for many performance measures, possibly negating the need for more complex numerical analysis. As such it is a very useful means of reducing the state space of large models without loss of accuracy.

Much work remains to be done in characterising models which are amenable to a quasi-separable solution. For instance where two actions of different types have equivalent behaviour can give rise to a much reduced model, as discussed in Section 5.2, whereas the characterisation of quasi-separability which has been considered here is based only on the types of actions which can occur.

The characterisation presented here is in an idealised form where components and environment are distinct. Work on product form decomposition has shown that such distinctions are not always clear and that the presence of such desirable solutions is rarely intuitive. It is expected that the classes of quasi-separable models will be similarly complex and that many further generalisations are possible. Cases when extensions to classes of models with product form solution give rise to quasi-separable models are also of practical significance.

Acknowledgements

The author wishes to thank Stephen Gilmore and Jane Hillston of University of Edinburgh for helpful comments during the continued development of this work and Prof. Isi Mitrani of University of Newcastle-upon-Tyne for introducing him to this area and for the initial insight which led to this approach.

References

[1] R.J. Boucherie, A characterisation of independence for competing Markov chains with application to Stochastic Petri nets, *IEEE Transactions on Software Engineering*, **20**(7), pp. 536–544, 1994.

[2] G. Clark, S. Gilmore, J. Hillston and N. Thomas, Experiences with the PEPA performance modelling tools, *IEE Proceedings - Software*, **146**(1), 1999.

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

[4] J. Hillston, Exploiting Structure in Solution: Decomposing Composed Models, in: C. Priami ed., *Proceedings of 6th International Workshop on Process Algebra and Performance Modelling*, 1998.

[5] J. Hillston and N. Thomas, Product Form Solution for a Class of PEPA Models, *Performance Evaluation*, to appear, 1999.

[6] I. Mitrani and P.E. Wright, Routing in the Presence of Breakdowns, *Performance Evaluation*, **20**, pp. 151-164, 1994.

[7] N. Thomas and S. Gilmore, Applying Quasi-Separability to Markovian Process Algebra, in: C. Priami ed., *Proceedings of 6th International Workshop on Process Algebra and Performance Modelling*, 1998.

[8] N. Thomas and I. Mitrani, Routing Among Different Nodes Where Servers Break Down Without Losing Jobs, in: F. Baccelli, A. Jean-Marie and I. Mitrani, eds., *Quantitative Methods in Parallel Systems*, pp. 248-261, Springer-Verlag, 1995.