Exploiting Structure in Solution: Decomposing Composed Models

Jane Hillston*

jeh@dcs.ed.ac.uk

Abstract

Since their introduction nearly ten years ago, compositionality has been reported as one of the major attractions of stochastic process algebras. The benefits that compositionality provides for model construction are readily apparent and have been demonstrated in numerous case studies. Early research on the compositionality of the languages focussed on how the inherent structure could be used, in conjunction with equivalence relations, for model simplification and aggregation. In this paper we consider how far we have been able to take advantage of compositionality when it comes to solving the Markov process underlying a stochastic process algebra model and outline directions for future work in order for current results to be fully exploited.

1 Introduction

Stochastic process algebras (SPA) were first proposed as a tool for performance and dependability modelling in 1989 [24]. At that time there was already a plethora of techniques for constructing performance models so the introduction of another one could have been deemed unnecessary if it were not for the fact that SPA offered something new—formally defined compositionality. Queueing networks, which have been widely used for performance modelling for more than thirty years, have an inherent compositionality but this is implicit and informal. Stochastic extensions of Petri nets have a semantic model but, in general, no clear compositional structure. In the process algebra the compositionality is explicit—provided by the combinators of the language—and formal—supported by the semantics and equivalence relations of the language.

It was immediately clear that having this explicit structure within models offers benefits for model construction:

- when a system consists of interacting components, the components, and the interaction, can each be modelled separately;
- models have a clear structure and are easy to understand;
- models can be constructed systematically, by either elaboration or refinement;
- the possibility of maintaining a library of model components, supporting model reusability, is introduced.

Several case studies demonstrating these and other benefits have appeared in the literature [25, 28, 46, 18, 32].

However, almost as quickly, it became clear that SPA models are prone to problems of state space explosion: making it easy for the modeller to represent systems in detail, coupled with the inherent complexity of the systems of interest, inevitably leads to models which are extremely large; in many cases, intractably so. In particular, coupled with the abstraction provided by the hiding combinator, compositionality allows a modeller to represent components of the system in detail, model their interaction in appropriate ways, and then abstract from the internal details of the combined component. This is a good technique for capturing the behaviour of systems. But note that although abstraction reduces the observability of actions, and in

*Laboratory for Foundations of Computer Science, The University of Edinburgh, Kings Buildings, Edinburgh EH9 3JZ. Tel: +44 131 650 5188.

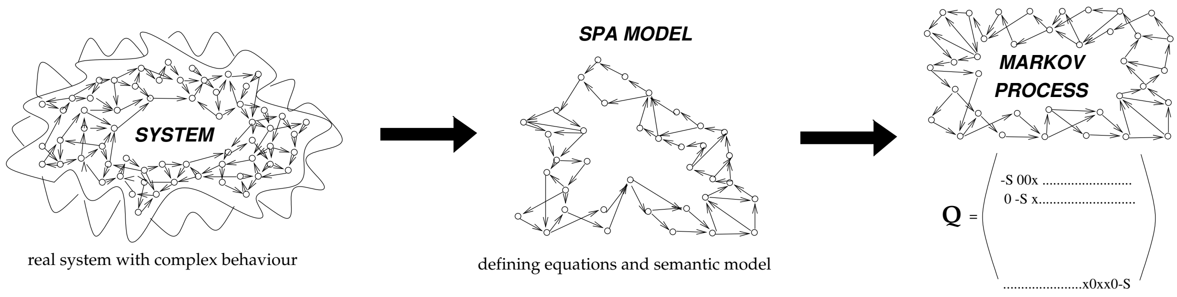

Figure 1: Schematic view of SPA modelling methodology

some languages reduces the measures that can be made on the model, it does not generally eliminate the internal states. Thus this attractive "feature" of SPA actually exacerbates the state space explosion problem.

Tackling this problem has been a major motivation of much SPA research for more than five years now. Of course, there are two state spaces which may be considered—the state space of the SPA model, which is generated in the labelled transition system via the operational semantics; and the state space of the underlying Markov process, the model to be solved (see Figure 1). Using the usual procedure for generating the underlying Markov process, there is an isomorphism between the two. We could try to attack the state space explosion problem at either level; indeed, there exists a substantial body of literature considering the problem of state space explosion at the level of the Markov process via a variety of techniques. But in order to benefit fully from the process algebra apparatus we choose to work at the level of the SPA.

Initial efforts concentrated on model manipulation techniques—model simplification and model aggregation. These techniques aim to *improve*, from the perspective of solution, the underlying Markov process, via manipulations of the state space at the SPA level. The most straightforward form of improvement is a reduction in the number of states. However, there are other possibilities, as we will discuss later in the paper. At the core of these techniques are equivalence relations, but compositionality also has an important role to play, as we will discuss in Section 2.

Unfortunately model manipulation alone still leaves us with a Markov process which must be solved numerically as a single entity. This leads to the inevitable question of how the compositional structure within SPA models relates to the various decomposed solution techniques which can be applied to Markov processes. Trying to answer this question has occupied several researchers recently and we survey this work in detail in Section 3. In general, work in this area has common elements although the approaches adopted by different researchers concentrate on different aspects of this general framework.

- *Characterisation of structures within the SPA model which correspond to decomposable structure in the underlying Markov process.* The aim of this work is to identify those SPA models which have a structure which is amenable to decomposed solution. In some cases this is fairly informal; in others the aim has been to establish syntactic rules which may be applied automatically, meaning that a tool may analyse a given model and decide whether it falls within the decomposable class or not.

- *Development of revised algorithms to generate the decomposed Markov process (usually as a set of Markov processes) from the SPA model.* Identifying the set of Markov processes susceptible to efficient solution is only the first step. In the "standard" solution algorithm there is a straightforward mapping from the semantic model to the state space of the underlying Markov process. If this process is to be decomposed, more sophisticated Markov process generation algorithms are needed. Once the composite Markov processes are formed, in some cases known solution algorithms can be applied; in some existing techniques can be modified; in others, new approaches to decomposed solution have been suggested by the process algebra structure. In the latter case, the new solution algorithms must also be developed.

- *Implementation of related algorithms.* Prototype tools and implementations of the new techniques, for characterisation, Markov process generation and decomposed solution, allow them to be tested in practice, and eventually, put to practical use.

Topics which have not yet received substantial attention but which are nevertheless interesting and important areas for future research are:

- *Establishing how performance measures may be specified in terms of the original model but calculated as accurately as possible in terms of the decomposed model.* In general, even when a naive approach to model solution is taken, based on numerical solution of the Markov process, the specification of the performance measures to be extracted from a SPA model is not wholly satisfactory. As yet little work has been undertaken to investigate how the compositional structure, and possibly decomposed solution, can be exploited for the calculation of performance measures.

- *Investigating how often "real" systems fall within the classes of model which are susceptible to efficient solution.* Aside from some work in [37], little experimental work has been done to establish how useful the techniques developed so far will be in practice. Unfortunately, in terms of direct application to models as produced by a modeller, the expectation is not high. However the real opportunity to harness the benefits of decomposed solution is likely to arise when the decomposition is used as a target for model manipulation.

- *Combining these techniques with model manipulation procedures.* As stated earlier, a smaller state space is not the only way in which we can think of a model being improved from the perspective of solution. Taking a model which is not obviously within a class which is susceptible to decomposed solution and manipulating it into a form where the decomposition is apparent, can be seen as an improvement of the model even if the size of state space is not reduced. This is a major area for future work and will be discussed in more detail in Section 4.

2 Harnessing compositionality

As outlined in the previous section, initial attempts to exploit the compositionality of SPA languages focussed on model manipulation—improving the model in some sense before the underlying Markov process is generated and solved.

There have been two principal approaches to model manipulation:

model simplification: Here an equivalence relation is used to establish behavioural or observational equivalence *between models*. The aim is to replace one model by an equivalent one which is more desirable from a solution point of view. Once the desirable model has replaced the original, the underlying Markov process is generated as usual, associating one state with each node in the labelled transition system generated by the semantics. Equivalence relations which have been used in this way are *weak isomorphism* in PEPA [25, 13], *Markovian bisimulation* and *weak bisimulation* in TIPP [37].

model aggregation: Here an equivalence relation is used to establish behavioural or observational equivalence *between states within a model*. The aim is to use an alternative mapping from the labelled transition system, given by the semantics of the model, to the underlying Markov process. The equivalence relation is used to partition the nodes of the labelled transition system into equivalence classes. Then, instead of the usual one-to-one correspondence between nodes and states, one state in the underlying Markov process is associated with each equivalence class of nodes. The hope is that this will generate a Markov process with a smaller number of states. The equivalence relation which has been used in this way is variously called *strong equivalence* (PEPA) [25], *Markovian bisimulation* (TIPP) [23], and *extended Markovian bisimulation equivalence* (EMPA) [3]. (For the remainder of this paper we will refer to it as *Markovian bisimulation*.)

In model aggregation the introduction of equivalence classes will generally reduce the number of states in the underlying Markov process and will certainly never increase it. Thus the resulting Markov process is more amenable to solution because its size has been reduced. Similarly, in model simplification reducing the number of states is the most straightforward way to make a model more desirable from a solution point of view; this is the approach taken in the work on weak isomorphism in PEPA. However there are other possibilities. For example, if the modified model falls within a class of models which are known to be amenable to an efficient non-standard solution procedure, the transformation will still aid solution even if the state space remains the same. This is the approach taken by Mertsiotakis's work on TIPP which will be discussed in more detail in Section 3.2.

Note that in general, when model manipulation is based on an equivalence relation which captures all relevant aspects of the behaviour of a model the subsequent solution of the transformed model will be exact [25, 23, 3]. However, when a partial order relation or an equivalence relation which does not consider all aspects of represented behaviour is used, model manipulation may result in an approximation of the original model. This is the case when weak bisimulation is used with respect to a subset of TIPP in Mertsiotakis's work on throughput approximation: the weak bisimulation relation cannot capture the timing characteristics of the models.

In the context of model manipulation, the role of compositionality is perhaps secondary to the equivalence relations which are used to define the model transformations. Nevertheless it is an important role, distinguishing the use of the model manipulation techniques in the SPA setting from their direct application at the Markov process level. Using the process algebra apparatus we are able to establish that the equivalence relations on which the transformations are based *respect the structure of the model*: by this we mean that the components which are explicit in the composite SPA model can be transformed individually and the modified model is formed as the composite of the modified components. This ability to use the equivalence relation in conjunction with the compositional structure relies on the equivalence relation being a *congruence*.

An equivalence relation is a congruence if it is *preserved* by the combinators of the algebra. For example, an equivalence relation, such as $\cong$, is preserved by a combinator, such as $\bowtie$, if whenever we know that $P \cong P'$ it follows that for any Q, $P \underset{L}{\bowtie} Q \cong P' \underset{L}{\bowtie} Q$. Once it is established that an equivalence relation is a congruence relation, it can be used in a manner complementary to the compositional structure of the model. Thus if we use the equivalence relation $\cong$ to transform model P to a more desirable form P', we can carry out the transformation of a composite, such as $P \underset{L}{\bowtie} Q$, component-by-component to form $P' \underset{L}{\bowtie} Q'$. Note that this is not necessarily the same as $(P \underset{L}{\bowtie} Q)'$ which may be even more desirable, but if $P \underset{L}{\bowtie} Q$ is too large to handle as a single entity, forming $P' \underset{L}{\bowtie} Q'$ may be "good enough". Alternatively we can apply the transformation to the improved model $P' \underset{L}{\bowtie} Q'$ since $(P' \underset{L}{\bowtie} Q')'$ will be equivalent to $(P \underset{L}{\bowtie} Q)'$.

Establishing that an equivalence relation is a congruence is something which is done once for any particular relation and set of process algebra combinators. In other words, as far as the performance modeller is concerned it is an established feature of the language. Moreover, the consequences of this for model manipulation are significant: as we have seen, components within the model may be manipulated, and *improved*, in isolation. Thus the state space of the complete model may never need to be constructed [25, 26, 23]. This greatly reduces the complexity of the procedure and ultimately, may make intractable models tractable.

Another approach which has been taken to exploit SPA model compositionality during Markov process generation, is the use of tensor algebra. Again, the objective is to tackle the problem of state space explosion but the strategy is to alter the representation of the underlying Markov process. Instead of capturing the Markov process as a single infinitesimal generator matrix, using tensor algebra it is represented as an expression of smaller matrices. Specially designed algorithms are able to take advantage of this expression and find the steady state solution in terms of the expression, avoiding the construction of the complete matrix. This general approach has been pioneered by Plateau and others with *Stochastic Automata Networks* [44, 45, 48].

In [10], Buchholz identifies the relationship between the parallel composition operator in the SPA language, MPA, and tensor algebra expressions for the underlying Markov process. He shows how an expression for the complete model can be constructed in terms of smaller matrices representing the individual components and the synchronisation sets in operation between them. Similarly, in [46], Rettelbach and Siegle construct a minimal compositional semantics for a subset of TIPP, called TIPP^{MS}. Specifically, this language includes a synchronising replication operator but not parallel composition in its general form. In this work a matrix is defined for each language expression without recourse to an operational semantics and the associated labelled transition system. This is achieved by constructing the matrix from sub-matrices corresponding to terms in the expression using matrix operators corresponding to the process algebra combinators. In particular, the replication operator maps to the tensor sum of replicated copies of the matrix corresponding to the replicated process.

We do not classify this work as decomposed solution since the underlying Markov process is still solved as a single entity although it is represented in a decomposed form. In the following section we survey work which we classify as truly decomposed solution. In these cases the SPA model is used to generate not one Markov process but several, and these processes are solved separately.

3 Decomposed solution

A variety of decompositional or structural techniques have been proposed to aid in the solution of large Markov processes. Recently several results have been published which show that, at least for some particular cases, there is a clear relationship between these techniques and the SPA model descriptions. In this section we survey these recent results.

In many cases the techniques which are applied are well-known at the Markov process level. The advantage of characterising the corresponding class of SPA models is that by "lifting" the definition from the stochastic process level to a formally defined high-level modelling paradigm we can facilitate the automatic detection of these structures when they occur, thus avoiding the construction of the original Markov process.

3.1 Product form solutions

It is clear that there is great advantage to be gained if the compositional structure of a SPA model can used during model solution, i.e. if the Markov processes corresponding to the components could be solved separately and their solutions combined to obtain a solution, exact or approximate, of the whole Markov process. One class of Markov processes 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.

Product form distributions have been widely used in the analysis of queueing networks and, due to their efficient solution, have contributed to the popularity of queueing networks for performance analysis. For example, Jackson networks [33] and their generalisation, BCMP-networks [2], have been widely employed. Here the underlying Markov process is known to have a reversible or quasi-reversible structure.

In contrast stochastic Petri nets (SPN) have rarely been found to be amenable to such efficient equilibrium solution, except when some of the expressibility of the formalism is reduced, for example by excluding resource sharing and competition over resources in a general form. By imposing these restrictions, Henderson and Taylor develop product form over the *places* of the Petri net, to obtain a product form similar to that obtained for queueing networks [21]. Lazar and Robertazzi establish a first step towards a product form over *subnets*, characterising independence between subnets which compete for resources [36]. Donatelli and Sereno show how both these approaches are related to T-semiflows in the Petri net [16].

Work on finding SPA models which give rise to product form solutions has drawn on the previous work on both queueing networks and SPNs, and a brief survey of this work is given in the paragraphs below. Essentially this can be seen as an investigation of when components *interact* and yet remain *statistically independent*. It is clear that when a SPA model consists of completely independent sequential components, i.e. $P \parallel Q$, the equilibrium distribution will have a product form:

$$\pi(P \parallel Q) = \pi_P(P) \times \pi_Q(Q) \tag{3.1}$$

where π_P and π_Q are the steady state distributions over the local states of P and Q respectively. However few real systems consist of components which are independent in this way, and if they did the state space explosion problem would not arise because it would be obvious that the components could be analysed separately. The challenge is to find circumstances in which components P and Q which synchronise, $P \underset{L}{\bowtie} Q$ in PEPA notation, still exhibit statistical independence.

Reversibility

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. At the Markov process level there are several ways to characterise these processes, but we state only the *local balance* condition. An irreducible, stationary Markov process $X(t)$ is *reversible* if it satisfies the detailed balance equations:

$$\pi(j)q(j,k) = \pi(k)q(k,j) \tag{3.2}$$

where $q(j,k)$ is the instantaneous transition rate from state j to state k and $\pi(\cdot)$ is the steady state probability distribution.

An initial study of SPA models giving rise to reversible Markov processes was presented by Bhabuta *et al.* in [4]. This paper largely considered the problem at the level of the underlying state space. In [30], Hillston and Thomas, identify syntactic conditions which a SPA 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 (those which do not involve any synchronisation) which give rise to reversible structures are identified. Then, assuming that a known class of reversible SPA components exist, the authors investigate under what circumstances the conditions for reversibility will be preserved if reversible components are composed using the combinators of the SPA.

Fundamental to the basic class of reversible sequential components is the notion of a *reverse pair*. A pair of action types $(\alpha, -\alpha)$ form a reverse pair if, in any state, any α transition leads to a state in which a $-\alpha$ transition leading back to the original state. This ability to "undo" any transition in the subsequent transition seems to be fundamental to reversibility. It clear to see that this is a necessary condition for equation 3.2 to be satisfied. From this starting pointing various canonical forms for sequential reversible components are described. The interesting conditions for when reversible components can be composed without losing the reversibility property relate to parallel composition when there is synchronisation or cooperation. In [30] detailed conditions, on the form of the synchronisation and the rates of the activities involved, are given. We refer the reader to that paper for more detail.

Quasi-reversibility

Like reversibility, *quasi-reversibility* is a type of product form originating from queueing theory. It is the condition which allows a wide class of queueing networks to be separated into their individual queues and solved in isolation, provided traffic equations are solved to give appropriate arrival rates at each queue. Formally, a stationary Markov process $X(t)$ is *quasi-reversible* if, for all times t_0 the state $X(t_0)$ is independent of

1. the input process after t_0 and
2. the output process before t_0.

Rather than the detailed balance equations which characterised reversibility, a quasi-reversible process satisfies *partial balance equations*:

$$\pi(i) \sum_{j \in S'} q(i,j) = \sum_{j \in S'} \pi(j)q(j,i) \tag{3.3}$$

for all states i and a corresponding subset of states S'. Again, more details of the definition of quasi-reversibility can be found in the excellent book by Kelly [35].

In [19], a SPA characterisation of this class is presented. As in the work on reversibility, the approach is to first find simple instances of SPA processes which give rise to quasi-reversible structure in their underlying Markov process. Then, conditions are established under which these components can be composed whilst maintaining the quasi-reversible property. Relative to the simple product form SPA case presented in equation 3.1, this does allow interaction between the components P and Q. But strong restrictions are placed on the form of this interaction. Again the notion of a *reverse pair* is important. Not surprisingly, given the origins in queueing networks, the form of admissible interaction is a *flow cooperation*. This means that the "positive" half of a reverse pair in one component is carried out in synchronisation or cooperation with the "negative" half of a reverse pair in another. The "positive" actions correspond to the input process in the definition of quasi-reversibility, while the "negative" correspond to the output process. The subsequent theorems reported in [19] correspond to those for open and closed queueing networks reported in [35].

Routing process approach

Sereno's work, reported in [47], derives product form criteria for SPA models based on earlier work on product form criteria for SPN [20, 21, 9]. 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. This condition places severe restrictions on the forms of synchronisation and resource contention which can be represented in the net.

In [47], Sereno uses a vector representation of the state of a SPA model in the characterisation of the class of models which have a product form based on the *routing process*. It is assumed that some preprocessing of the model is done in order to collect information and to aggregate the model, using one of the techniques outlined in Section 2 of this paper. The information which is needed is the local state space of each component, and, for each action type of the model, which local states of participating components enable the action and which appear after it has been performed. These two sets of local states are called the *pre-set* and *post-set* of the action, respectively. The state vector representation is composed of sub-vectors, one corresponding to each defined sequential component; an element within the sub-vector corresponds to a local state within corresponding component. In the representation of any particular state the value of an element within the vector records the number of instances of each local state exhibited in the current syntactical state of the model. Storing the state in this form, together with the pre- and post-sets of actions represented as vectors, allows the effect of completing an action to be written down in vector form.

There are several restrictions placed on the SPA models, in particular with respect to action types. An action can only have one pre-set—this implies that each action within the behaviour of a sequential component must have a distinct name. Moreover if actions of the same type occur within different sequential components they must be synchronised.

Sereno's approach for SPA is completely analogous to the earlier work on SPN—he defines a Markov chain in which the states correspond to the actions of the SPA model. This is called the *routing process*. The global balance equations of the routing process correspond to the traffic equations of queueing networks. If the state space of this process can be partitioned into equivalence classes of enabling actions (roughly speaking, one action *enables* another if the post-set of one is the pre-set of the other; we take the transitive closure of that relation), then a product form solution exists. Moreover the partition forms the basis for the decomposition.

Product form over submodels

As mentioned earlier, in [36] Lazar and Robertazzi investigate a product form in the context of SPN—the decomposition is carried out over subnets, which may still need to be solved by standard numerical techniques to find their local steady state.

In [8], Boucherie generalised their result and characterised the class of underlying Markov processes. Such a process is formed as the product of a set of Markov processes which compete over a set of resources. The resources are not explicitly represented but the competition has two important impacts on the state space and the transition rates of the product process. Firstly, if two constituent processes compete over a resource they cannot both enter the region of their state space representing possession of the resource at the same time. Thus areas of the state space of the product process are eliminated. It is assumed that the product process will change state in only one of the constituent processes at each state change. The second effect of the competition over resources is to limit this still further—if two processes compete over a resource, and one of them is currently holding the resource, then the other cannot make any state change, no matter where it is in its state space. Thus when a constituent process holds a resource, all competing processes are blocked. Essentially the product form result holds because in each state of the product process each constituent process satisfies its own global balance equations. If it can make a transition it is free to act as if it were independent; alternatively, it is completely blocked and satisfies its global balance equations trivially.

In [27, 31], Hillston and Thomas aim to characterise this class of Markov processes in the SPA language PEPA. The class of models that they identify consist of independent components, which give rise to the constituent Markov processes of the underlying Markov process. These components interact indirectly by synchronisation with resource components. Compared with the simple product form model presented in equation 3.1, the general form of these process algebra terms and the resulting product form is, schematically:

$$\pi\left((P \parallel Q) \underset{L}{\bowtie} R\right) = B \times \pi_P(P \underset{L}{\bowtie} R) \times \pi_Q(Q \underset{L}{\bowtie} R) \tag{3.4}$$

where the component R represents the resource, π_P and π_Q are the steady state distributions over the derivatives of $P \underset{L}{\bowtie} R$ and $Q \underset{L}{\bowtie} R$ respectively, and B is the normalising constant. The decomposition is formed by considering each of the model terms (P and Q in this case) acting in synchronisation with the resource (R) in isolation.

In the SPA, a component is termed a *resource* if it is never free to act independently; all its activities must be carried out in synchronisation with the rest of the model. All components are assumed to have cyclic behaviour. In this context a component is considered to be using or holding the resource if it has carried out the first action of the resource's behaviour in synchronisation with the resource. The semantics of the SPA ensure that the state space of the product process is suitably modified, i.e. that two competing components cannot hold the resource simultaneously. In order to ensure that the correct condition is also satisfied by the transition rates of the product process, Hillston and Thomas place a further restriction on the synchronisation set in operation between the resource component and the rest of the model. If a model component wishes to use the resource during one of its possible behavioural cycles, it must gain control of the resource at the start of the cycle and release it only at the end. This guarantees that other competing components will be blocked when the component holds the resource. Although presented here informally, these conditions are defined as formal syntactical conditions which can be checked on the model specification.

As we have seen, the previous work on product form SPA models centred on components of a particular structure which interact in a restricted way, preserving a form of independence between the components. Here the characterised models represent the competition of otherwise independent components over resources. The form of these components is not restricted; however they do place a relative restriction on the form of the resource and on the form of the cooperation set in operation between the resource and the rest of the model. As suggested by equation 3.4 above, these components, together with the resource, are solved in isolation, these partial solutions subsequently being combined to give a solution of the complete model. The outstanding problem of this approach, however, is the calculation of the normalising constant.

The models presented in [8], including 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 SPN models, a non-standard state representation had to be used in order to eliminate the resource from the model representation. The SPA models do not have this disadvantage since the resource may (indeed, must) be represented explicitly and subsequently eliminated from the state representation using formally defined procedures.

Quasi-separability

A quasi-separable Markov process does not have a product form solution in the sense of the other classes of models considered in this section. However we discuss this approach here because it has more similarities with the product form cases than with the aggregated decomposed solutions considered in the following section. Unlike the case with product form processes it is not possible to find the exact solution of the steady state distribution of a quasi-separable process as a product of the local steady state distributions. Nevertheless decomposed solution can lead to exact results for the local steady state distributions and many performance measures, and no aggregated model needs to be formed.

As with reversibility and quasi-reversibility, the notion of quasi-separability is one which has been developed in relation to queueing networks, in particular queueing networks in which breakdowns occur [42, 41]. It is assumed that the Markov process is comprised of a number of components and that there are two pertinent pieces of information for each component. A representation of the whole process can then be formed as a pair of vectors, each vector capturing one piece of information for each component. For a process to be *quasi-separable* it must be possible to analyse the behaviour of a component, say component i, given the ith element of the first vector and all elements of the second, or vice versa. This allows the complete process to be reduced to a number of sub-models, each of which contains all the information about exactly one component. In the queueing network context the two pieces of information about each queue are typically its operational state and the number of customers present.

In [49], Thomas and Gilmore present a characterisation of SPA models which are quasi-separable. It is assumed in this characterisation that the information which must be included in each decomposed submodel is not distributed between the components but maintained by a single *scheduler* component. There are several conditions on the way in which this component may interact with the other components of the model, which do correspond to the components of the system. When the scheduler changes its state it must do so by an individual action, or by a shared action in which the other participant is passive. Furthermore the individual components have no direct interaction between them—they must be in parallel composition with no synchronisation. In other words, each of these components interacts only with the scheduler. In some senses this is similar to the scenario for the product form over submodels described above, but note that the behaviour of the scheduler and the resource, in relation to the rest of the model, are quite different. The model is decomposed into a set of models, each comprising of a single component considered with the scheduler, in isolation. More details can be found in [49].

In all the cases reported above, the primary focus has been on characterising SPA models which, when the semantics of the language are applied and the labelled transition system formed, generate Markov processes within the given class. The aim is to develop the characterisation as a set of syntactical conditions which can be tested without having to apply the semantics to the whole model, although investigation of the state space of individual components may be necessary. Moreover, these conditions should be sufficiently formal that they can be incorporated into one of the SPA tools, such as the PEPA Workbench [17], allowing the recognition of the structure to be automated. Note that in each case, the current characterisation is known to be incomplete in the sense that there are SPA models which give rise to Markov processes of the appropriate class which would not be recognised by the current conditions. Extending the characterisations is on-going work.

3.2 Aggregated decomposed solutions

In this section we consider *aggregated decomposed* solutions. Here, it is not simply a case of splitting the model into submodels or components in the style of product form. As well as a stochastic representation of each of the components, the decompositional solution involves a stochastic representation of the interactions between these components, the *aggregated model*. In most cases these stochastic representations will be Markov processes but in the work by Bohnenkamp and Haverkort on decomposition via synchronisation points semi-Markov processes are used [7]. This paper is also the exception in not having been inspired by earlier work on SPN. It may be the first example of a decomposition technique being suggested by the SPA model structure.

Time scale decomposition

The work on time scale decomposition in SPA is based on the notion of *near completely decomposable* Markov processes [14], and inspired by previous work on time scale decomposition of SPN models [5, 1]. A characterisation of a near completely decomposable Markov process at the matrix level is that the matrix is block structured with elements in the diagonal blocks being at least an order of magnitude larger than elements in the off-diagonal blocks. This implies that the model is made up of subsystems whose internal interactions are much more frequent than the interactions between subsystems. As a consequence it can be assumed that the subsystems reach an internal equilibrium between external interactions. Thus a steady state for each Markov process corresponding to a diagonal block of the original process is found; the interactions are modelled by an aggregated model capturing the interactions between subsystems as represented by the off-diagonal blocks. The aggregated model has one state for each subsystem/diagonal block. There are known error bounds for the technique based on the magnitude of the largest element in an off-diagonal block.

The initial classification of SPA models susceptible to time scale decomposition [29], relied on a classification of the sequential components within a model into *fast* or *slow*; this in turn was based on a classification of all actions relative to some threshold rate. A component is considered to be fast if it enables fast or passive actions; a component is considered to be slow if it enables only slow actions. Only models comprised of fast and slow components were considered. The state of such a process may be represented as the vector of local states for each sequential component: this is called the *state vector*. So a model P which is comprised of k fast and ℓ slow components may be represented as: $P \equiv (F_1, \ldots, F_k, S_1, \ldots, S_\ell)$. Then each decomposed component corresponds to a set of state vectors which exhibitthe same derivatives in all the slow components:

$$A_{[S_1,\ldots,S_\ell]} \equiv \{(F_1', \ldots, F_k', S_1', \ldots, S_\ell') \mid S_1' \equiv S_1, \ldots, S_\ell' \equiv S_\ell\}$$

The elements of this set are found using the semantics of the language when the original model is considered in composition over a synchronisation set which blocks all the slow actions. Finding other decomposed components, and constructing the aggregated model, is achieved by removing this blocking synchronisation and allowing the current submodel to evolve just one step by a slow action. Note that the aggregated model does not have a representation at the SPA level, only as a Markov process.

Later work by Mertsiotakis [38, 37], tackles the problem of *hybrid* components—these are sequential components which cannot be classified as either fast or slow since they enable both fast and slow actions. Rather than split a hybrid component into a fast and slow component, it was decided to extract the slow behaviour of the hybrid into a separate *shadow* component, making the original component passive with respect to these actions. In [37] it is shown that such a transformation preserves the behaviour of the hybrid component, and, since the equivalence relation is a congruence, the behaviour of the complete model. Once the transformation has been completed the original procedure can be applied.

Decision free processes

Mertsiotakis and Silva's work on decomposition of a class of SPA models, termed *decision free processes*, is based on earlier work on throughput approximation in a class of SPN called *marked graphs* [34]. Essentially, the idea is to partition the model into components, typically two in the marked graph case. Decomposed models are then formed in which one component is fully represented while the other is reduced to a minimal form, usually consisting of a single transition. In addition to these two decomposed models there is also an extremely simple aggregated model, consisting of the two minimal forms linked appropriately. An iterative scheme is then used to find a solution to the model, the influence of one component on the other being represented in the decomposed model by the rate of the transition in the minimal form.

The decision free process approach to throughput approximation [39, 40, 37] relies on the decomposition of a decision free process into three components, one of which acts as an intermediary between the other two. This component is distinguished as the *interface*. Note that the components do not necessarily correspond to sequential components (c.f. time scale decomposition). The decomposed Markov processes are generated from the consideration of the two possible (component, interface) pairs. In each case a reduced representation of the interface is used, corresponding to this component's view. In addition a basic skeleton is formed which corresponds to a greatly reduced version of the complete model, in which only the interface actions are carried out. Once this decomposition has been carried out, the algorithm follows the same general form as outlined above for the marked graph case.

Rather than a characterisation to recognise models of this form, this work relies on models having been constructed in the specified way. The class of decision free processes is defined via a reduced syntax, disallowing the choice combinator, +, and placing restrictions on where action types may appear within components. In particular any action may be performed only once within any cycle of behaviour. This condition removes the possibility of implicit choice, when the action is to be carried out in synchronisation with another component. It is recognised that even working within this class the necessary structure, with an interface component acting as intermediary between two other components, may not be immediately apparent within the model. Therefore a series of possible transformations are defined, each of which is shown to be based on an equivalence relation which preserves some aspects of the model's behaviour. However, note that for two of the transformations the equivalence considers only functional, not temporal, behaviour. Thus it guarantees, for example, that no deadlock is introduced into the model but tells us nothing about the timing characteristics of the new model in relation to the old one.

Near-independence

In [12], Ciardo and Trivedi present a decomposition technique for stochastic reward nets (a version of SPN with immediate transitions, inhibitor arcs and rewards) based on the notion of *near-independence*. Components are considered to be near-independent if they operate in parallel and only rarely interact. The basic idea is that near-independent components can be solved independently in conjunction with a graph model which represents the (limited) dependencies that do still exist between them. Several examples of canonical near-independent net structures are given in the paper, but in general recognising such structures in any given model, and whether necessary conditions on the graph model are met, rely on the expertise of the modeller.

Components are solved in isolation, as if they were independent, but their influence upon each other is approximated by the rate at which synchronisation can take place. This is estimated using the dependency graph. In general, fixed point iteration may be necessary in order to achieve a satisfactory solution of the complete model, depending on the structure of the graph.

In [6], Bohnenkamp and Haverkort suggest that this technique could be adapted for SPA models. This paper does not progress this directly in terms of an SPA language but does report some interesting experiments which investigate the feasibility of the approach. In the proposed approach the dependence between components is recognised as the actions on which they synchronise (synchronisation between delays is not allowed). In effect the behaviours of the near-independent components are serialised, first capturing the work which can be done until blocking occurs due to a synchronisation point and then the work necessary to achieve the synchronisation.

Decomposition via synchronisation points

In [7], Bohnenkamp and Haverkort develop the ideas from [6] in a slightly different direction. The approach still centres on the role of synchronisations between parallel components, but now the aim is to reformulate the underlying Markov process in terms of a set of semi-Markov processes. These semi-Markov processes are solved via their embedded Markov chains and evaluations of the distribution of the times between synchronisation points. Working within a SPA framework, they consider a class of models in which there is a fixed number of sequential processes composed in parallel, assuming each composition is subject to the same set of global synchronisation actions. Within this class of models their solution technique is exact with respect to throughputs and local steady state probabilities.

In the original SPA languages a delay is associated with each action representing its duration, resulting in a labelled transition system in which arcs are labelled by two types of information: action type and rate information. All the work already described in this paper is based on such languages. However, Bohnenkamp and Haverkort use a language in which actions and time delays are treated separately. Several recent SPA languages [22, 15] take this approach, which is also found in the timed process algebras. Here the transition system has two distinct transition relations: one representing instantaneous actions and the other representing the passing of (stochastic) time.

From the point of view of the work reported in [7], this simplifies somewhat as only actions are allowed to synchronise, and the authors do not need to be concerned about the meaning of synchronisation between two delays. The sequential components of the model are treated as the decomposed processes of the underlying Markov process. The aggregated model, the embedded Markov chain of a semi-Markov process, is constructed compositionally: a tensor expression is formed from the embedded Markov chain of the semi-Markov process corresponding to the synchronisation process of each sequential process algebra component. This EMC may have several disjoint components but the initial state of the process is used to ensure that only the "live" component is considered. The reader is referred to [7] for more details.

4 Conclusions and future work

Clearly, decomposed solution of SPA models is possible. Moreover, it can be achieved by exploiting structures which are introduced at the process algebra level, to elucidate structures in the underlying Markov process. Of course, the class of models which can be recognised and handled by each of the current techniques, reported in the previous section, is somewhat limited. However the diversity of these techniques means that together they represent a substantial class of models.

Although the current approaches to decomposed solution differ, there are some common points from which we can perhaps learn:

- In general, interaction between components is the major barrier to exact decomposition (i.e. product form solution), although it holds the key to some of the aggregated decompositional techniques. There is scope to review the combinators for interactions which are offered by the SPA languages. Perhaps it would be possible to define a combinator which offered a restricted form of interaction which maintained statistical independence, such as the flow cooperation for quasi-reversible PEPA models.

- In the characterisations which have been carried out some conditions seem to appear with regularity: for example, the condition that an action type may occur only once in the definition of a sequential component. The implications of imposing such conditions on *all* models should be investigated.

- Inspiration from older paradigms, especially SPN is clearly important, as can be witnessed by the number of approaches reported in this paper which have developed via that route. However, it should be made clear that in most cases this is *inspiration* and not *translation*—if the technique is to work well within the new framework it should take full advantage of the process algebra apparatus. In some cases, such as time scale decomposition [29, 37], the resulting algorithm is actually simpler because we can take advantage of the SPA semantics.

There is much work still to be done. An interesting project would be to compare the published case studies with the published classes of "well-behaved" models. Although this would not advance the theory it would give us some impression of how useful it is. Unfortunately the overlap between case studies and characterisations is likely not to be as substantial as we would like. This however leads on to another promising direction for future work: the development of model manipulation techniques which are complementary to the decomposed solution techniques.

The basic idea is that once a model has been constructed, with tool assistance, the modeller will be able to massage her model into one of the classes of models corresponding to a decomposed solution technique.

In the most straightforward case this would be based on one of the existing equivalence relations, using rewriting rules. For any SPA model there are often several equivalent ways in which it can be expressed. The characterisations often assume one particular form which may not be readily apparent. For example, a PEPA model $(P \underset{K}{\bowtie} R) \underset{L}{\bowtie} Q$ may be equivalent to $(P \parallel Q) \underset{K \cup L}{\bowtie} R$. In the first form it is not a candidate for the product form over submodels while in the second form it is.

In general, however, such non-intrusive manipulations will not be sufficient. The model will not belong to a class with efficient solution techniques. The aim will be to transform it into a similar model which is; moreover to carry out that transformation formally. Building on the formal semantics of the SPA, this manipulation will be carried out subject to established partial order relations which guarantee that for the performance measure of interest, the new model performs at least as well as the original model. Thus the new model can be regarded as a bound for the original model. Note that we anticipate that such partial order relations will have to be performance measure specific. Also, that these will result in a procedure for which there is tool-assistance but not necessarily automation.

Developing such partial order relations is a major undertaking. There is some previous work from timed process algebras[43] and from SPN[11] which we can draw upon but largely this is uncharted territory. The rewards, however, will be considerable.

It is the prerogative of an invited paper to pose at least as many questions as it answers and in the area of efficient solution of the Markov processes underlying SPA models there are many questions still to be answered. However the work that has been completed so far demonstrates that *automated* decomposed solution of SPA models is feasible. As such, it offers a solution, albeit partial, to the state space explosion problem. Moreover, in several cases this solution can already be applied transparently to the modeller [37]. This represents a significant step forwards for performance modelling using process algebras.

Acknowledgements

Jane Hillston's work is supported by the Engineering and Physical Sciences Research Council via the COMPA project (G/L10215). I would like to thank my colleagues who have contributed the interesting work reported in Section 3. I would like to apologise to those whose work I did not include: this was due to the limited space available. Finally, my thanks to Graham Clark, Stephen Gilmore and Marina Ribaudo who read earlier drafts of this paper.

References

[1] H.H. Ammar and S.M. Rezaul Islam. Time Scale Decomposition of a Class of Generalized Stochastic Petri Net Models. *IEEE Transactions on Software Engineering*, 15(6):809–820, June 1989.

[2] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2):248–260, April 1975.

[3] M. Bernardo and R. Gorrieri. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. *Theoretical Computer Science*, to appear, 1998.

[4] M. Bhabuta, P.G. Harrison, and K. Kanani. Detecting reversibility in Markovian Process Algebra. In *Performance Engineering of Computer and Telecommunications Systems*, Liverpool John Moores University, September 1995. Springer-Verlag.

[5] A. Blakemore and S. Tripathi. Automated Time Scale Decomposition of SPNs. In *Proc. of 5th Int. Workshop on Petri Nets and Performance Models (PNPM '93)*, Toulouse, 1993.

[6] H. Bohnenkamp and B. Haverkort. Decomposition Methods for the Solution of Stochastic Process Algebra Models: a Proposal. In E. Brinksma and A. Nymeyer, editors, *Proc. of 5th Process Algebra and Performance Modelling Workshop*, 1997.

[7] H. Bohnenkamp and B. Haverkort. Semi-Numerical Solution of Stochastic Process Algebra Models. In C. Priami, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, 1998.

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

[9] R.J. Boucherie and M. Sereno. On the Traffic Equations of Batch Routing Queueing Networks and Stochastic Petri Nets. Technical report, European Research Consortium for Informatics and Mathematics, 1994.

[10] P. Buchholz. Compositional Analysis of a Markovian Process Algebra. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[11] J. Campos, B. Sanchez, and M. Silva. Throughput Lower Bounds for Markovian Petri Nets: Transformation Techniques. In *Proc. of the 4th Int. Workshop on Petri Nets and Performance Models*, pages 322–331. IEEE Computer Society Press, December 1991.

[12] G. Ciardo and K.S. Trivedi. A Decomposition Approach for Stochastic Petri Net Models. *Performance Evaluation*, 1992.

[13] G. Clark. An Extended Weak Isomorphism for Model Simplification. In E. Brinksma and A. Nymeyer, editors, *Proc. of 5th Process Algebra and Performance Modelling Workshop*, 1997.

[14] P.J. Courtois. *Decomposability: Queueing and Computer System Applications*. ACM Series. Academic Press, New York, 1977.

[15] P. D'Argenio, J-P. Katoen, and E. Brinksma. General Purpose Discrete Event Simulation Using Spades. In C. Priami, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, 1998.

[16] S. Donatelli and M. Sereno. On the Product Form Solution for Stochastic Petri Nets. In *Application and Theory of Petri Nets*, pages 154–172. Springer Verlag, 1992.

[17] 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, editors, *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 794 of *LNCS*, pages 353–368. Springer-Verlag, 1994.

[18] S. Gilmore, J. Hillston, R. Holton, and M. Rettelbach. Specifications in Stochastic Process Algebra for a Robot Control Problem. *International Journal of Production Research*, December 1995.

[19] P. Harrison and J. Hillston. Exploiting Quasi-reversible Structures in Markovian Process Algebra Models. *The Computer Journal*, 38(6), 1995. Special Issue: Proc. of 3rd Process Algebra and Performance Modelling Workshop.

[20] W. Henderson, D. Lucic, and P.G. Taylor. A Net level Performance Analysis of Stochastic Petri Nets. *Journal of the Australian Mathematical Society, Series B*, 31:176–187, 1989.

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

[22] H. Hermanns and M. Rettelbach. Towards a Superset of Basic LOTOS for Performance Prediction. In M. Ribaudo, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, 1996.

[23] H. Hermanns and M.L. Rettelbach. Syntax, Semantics, Equivalences and Axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[24] U. Herzog. Formal description, time and performance analysis: A framework. Technical Report 15/90, IMMD VII, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany, September 1990.

[25] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, Department of Computer Science, University of Edinburgh, April 1994. CST-107-94.

[26] J. Hillston. Compositional Markovian Modelling Using a Process Algebra. In W.J. Stewart, editor, *Numerical Solution of Markov Chains*. Kluwer, 1995.

[27] J. Hillston. A Class of PEPA Models Exhibiting Product Form Solution. Technical Report ECS-LFCS-98-382, LFCS, Dept. of Computer Science, University of Edinburgh, February 1998.

[28] J. Hillston, H. Hermanns, U. Herzog, V. Mertsiotakis, and M. Rettelbach. Integrating Qualitative and Quantitative Modelling with Stochastic Process Algebras. Technical report, IMMD VII, Universität Erlangen-Nürnberg, May 1994.

[29] J. Hillston and V. Mertsiotakis. A Simple Time Scale Decomposition Technique for Stochastic Process Algebras. *The Computer Journal*, 38(6), 1995. Special Issue: Proc. of 3rd Process Algebra and Performance Modelling Workshop.

[30] J. Hillston and N. Thomas. A Syntactical Analysis of Reversible PEPA Models. In *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998. University of Verona.

[31] J. Hillston and N. Thomas. Product Form Solution for a Class of PEPA Models. In *Proc. of IEEE International Computer Performance and Dependability Symposium*, Durham, NC, September 1998. IEEE Computer Society Press.

[32] D.R.W. Holton. A PEPA Specification of an Industrial Production Cell. *The Computer Journal*, 38(6), 1995. Special Issue: Proc. of 3rd Process Algebra and Performance Modelling Workshop.

[33] J.R. Jackson. Jobshop-like Queueing Systems. *Management Science*, 10:131–142, 1963.

[34] H. Jungnitz. *Approximation Methods for Stochastic Petri Nets.* PhD thesis, Rensselaer Polytechnic Institute, May 1992.

[35] F. Kelly. *Reversibility and Stochastic Processes.* Wiley, 1979.

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

[37] V. Mertsiotakis. *Approximate Analysis Methods for Stochastic Process Algebras.* PhD thesis, Universität Erlangen–Nürnberg, Martensstraße 3, 91058 Erlangen, April 1994.

[38] V. Mertsiotakis. Time Scale Decomposition of Stochastic Process Algebra Models. In E. Brinksma and A. Nymeyer, editors, *Proc. of 5th Process Algebra and Performance Modelling Workshop*, 1997.

[39] V. Mertsiotakis and M. Silva. A Throughput Approximation Algorithm for Decision Free Processes. In M. Ribaudo, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, 1996.

[40] V. Mertsiotakis and M. Silva. A Throughput Approximation Algorithm for Decision Free Processes. In M. Ribaudo, editor, *Proc. of 7th Int. Workshop on Petri Nets and Performance Models*, 1996.

[41] I. Mitrani and N. Thomas. Routing Among Different Nodes Where Servers Break Down Without Losing Jobs. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 248–261. Springer, 1995.

[42] I. Mitrani and P.E. Wright. Routing in the Presence of Breakdowns. *Performance Evaluation*, 20:151–164, 1994.

[43] F. Moller and C. Tofts. Relating processes with respect to speed. Technical Report ECS-LFCS-91-143, LFCS, Department of Computer Science, University of Edinburgh, January 1991.

[44] B. Plateau. On the Stochastic Structure of Parallelism and Synchronisation Models for Distributed Algorithms. In *Proc. ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems*, 1985.

[45] B. Plateau, J.M. Fourneau, and K.H. Lee. PEPS: A Package for Solving Complex Markov Models of Parallel Systems. In *Proc. of the 4th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, 1988.

[46] M.L. Rettelbach and M. Siegle. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[47] M. Sereno. Towards a Product Form Solution of Stochastic Process Algebras. *The Computer Journal*, 38(6), 1995. Special Issue: Proc. of 3rd Process Algebra and Performance Modelling Workshop.

[48] W.J. Stewart, K. Arif, and B. Plateau. The numerical solution of Stochastic Automata Networks. *European Journal of Operation Research*, 86(3):503–525, 1995.

[49] N. Thomas and S. Gilmore. Applying Quasi-Separability to Markovian Process Algebra. In *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998. University of Verona.