Process Algebras for Quantitative Analysis

J. Hillston
LFCS, School of Informatics
University of Edinburgh, UK
jeh@inf.ed.ac.uk

Abstract

In the 1980s process algebras became widely accepted formalisms for describing and analysing concurrency. Extensions of the formalisms, incorporating some aspects of systems which had previously been abstracted, were developed for a number of different purposes. In the area of performance analysis models must quantify both timing and probability. Addressing this domain led to the formulation of stochastic process algebras. *In this paper we give a brief overview of stochastic process algebras and the problems which motivated them, before focussing on their relationship with the underlying mathematical stochastic process. This is presented in the context of the PEPA formalism.*

1 Introduction

The development of stochastic process algebras (SPAs) was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes. Performance analysis seeks to predict the behaviour of a system with respect to dynamic properties such as the number of requests that can be satisfied per unit time (throughput) and response time.

There are a variety of approaches available for the performance evaluation of systems. If the system exists it may be possible to monitor the system directly. However, in general, such an approach is time-consuming, difficult and lacks generality. Therefore it is often preferable to model the system rather than use such direct experimentation. Indeed, when the system is yet to be constructed, modelling is the only option.

Performance models may be analysed by simulation, numerical solution or analytical solution. Simulation models have the advantage of being insensitive to state space size. Unfortunately such models are time-consuming to analyse and bring the intellectual burden of evaluating the trustworthiness of results by the calculation of confidence intervals.

In contrast analytic solution — in which an expression for the performance measure of interest is derived in terms of the input parameters of the model — can be extremely efficient to use. However, constructing such solutions is very much the domain of the expert and typically each system requires a bespoke solution.

Numerical solution of a Markov chain offers a compromise between these two extremes. Some assumptions about the system are needed, particularly with respect to the timing of events. But the resulting models are relatively straightforward to solve, relying only on simple linear algebra techniques. For moderately sized models the generator matrix of the Markov chain can be stored as a dense matrix, admitting direct solution methods with good numerical accuracy. For larger models sparse matrices are needed, necessitating the use of iterative solution techniques with some loss of numerical precision. The largest models require yet more ingenuity in the representation of the matrix using Kronecker or BDD-based storage.

Markov processes (sometimes termed Continuous Time Markov Chains (CTMCs)) whilst offering general applicability, are difficult to construct for large systems so an intermediate system description language is often used. In the early nineties the most common of these were *queueing networks* [41] and *Stochastic Petri Nets* (SPN) [46]. Queueing networks, whilst very powerful when applicable, have limited expressiveness and lack formal interpretation. SPN models have formal interpretation but do not have the explicit structure found in queueing networks, which greatly eases model construction.

Researchers in performance analysis were attracted to process algebras for a number of reasons.

Compositionality: The ability to construct models in a compositional manner is a significant benefit when large complex systems are under consideration. Furthermore, it was already established that for qualitative properties analysis could also be compositional.

Qualitative analysis: For many systems it is important to verify both correct functionality and timeliness of response. However, there was an issue of consistency when different formalisms and different techniques were being used for modelling for these two objectives. Using an elaborated version of a functional modelling technique meant that the same system description could be used to project models for both purposes.

Wide acceptance: Process algebras were perceived to have gained some acceptance outside academia in the form of languages like LOTOS [37]. There was, and still is, a problem that performance analysis is often neglected until late in the development cycle when problems become apparent. The impact can be that compromises are needed on original performance requirements or that substantial and expensive re-designs are needed.

In this paper we discuss the extent to which these initial expectations have been met, focusing particularly on compositionality. The rest of the paper has the following structure. In Section 2 we present the principles of stochastic process algebra and the PEPA language in particular. Issues of compositionality are discussed in Section 3, focusing on the interplay between the process algebra and the underlying mathematical model. Integrated analysis and accessibility are reviewed in Sections 4 and 5 respectively. Finally, in Section 6 we draw some conclusions.

2 Stochastic Process Algebra

In order to carry out performance analysis of a system, it is essential to record information about the timing characteristics of the system and the relative probabilities of alternative behaviours. Without this quantified information it is not possible to derive quantitative measures such as expected response time or throughput. Hardware details are generally abstracted and therefore a continuous time rather than a discrete time model is appropriate for most performance studies. Furthermore, the abstraction of data aspects of the system, and the involvement of human users, make exact timings unpredictable.

Therefore in order to create a process algebra suitable for performance modelling the quantification of the model is achieved using random variables. In the initial calculi a random variable was associated with each of the actions of the model, specifying the delay incurred when the action is performed [27]. Different languages had different target analysis techniques and consequently chose to specify this delay differently. For example, early versions of TIPP [21] used execution traces and allowed general distributions, as did SPADES [49] which was intended for discrete-event simulation. PEPA, which was intended as a high-level description language for Markov processes, was the first SPA to restrict the duration of activities to be governed by negative exponential distributions [31].

It is known that the only distribution for state sojourn times which gives rise to the Markov property is the negative exponential distribution (which has distribution $F = 1 - e^{-\lambda t}$, meaning that the probability to leave the state before time t is $1 - e^{-\lambda t}$).

Definition 2.1 (Markov process) *A stochastic process $X(t), t \in [0, \infty)$ with discrete state space $\mathcal{S}$ is a* Markov process *if and only if, for $t_0 < t_1 < \ldots < t_n < t_{n+1}$, the joint distribution of $(X(t_0), X(t_1), \ldots, X(t_n), X(t_{n+1}))$ is such that*

$$\Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_0) = s_{i_0}, \ldots X(t_n) = s_{i_n})$$
$$\Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_n) = s_{i_n})$$

Subsequent versions of TIPP also adopted exponential delays associated with actions [28], and the language EMPA generalised this to allow the possibility of instantaneous actions, whilst still giving rise to a Markov process [6]. Quantification is added to the stochastic π-calculus in the same manner as PEPA [48].

Later languages separated time and action introducing distinct forms of prefix to capture the two forms of evolution. This was done in the Markovian setting in IMC [25] and in more general stochastic setting in ♠ [16], IGSMP [11] and Modest [17] where time evolution may be specified with general distributions. This work was complemented by algebras which sought to investigate the cases in which general distributions could be incorporated into languages which maintained the integrated view of delay and activity such as GSMPA [10] and Clark's work on a generally distributed extension of PEPA [13]. In most cases such languages must rely on simulation for evaluation purposes.

2.1 PEPA

As explained above, PEPA was developed as a high-level description language for Markov processes. Thus it extends classical process algebra by associating a negative exponentially distributed random variable, representing duration, with every action. There is no explicit probabilistic choice operator, but an implicit choice is associated with each choice by the assumption of the *race condition*. This leads to a clear relationship between the process algebra model and a Markov process. Via this underlying stochastic process performance measures can be extracted from the model.

PEPA models are described as interactions of *components*. Each component can perform a set of actions: an action $a \in \mathcal{A}ct$ is described by a pair (α, r), where $\alpha \in \mathcal{A}$ is the *type* of the action and $r \in \mathbb{R}^+$ is the parameter of a negative exponential distribution governing its duration.

Whenever a process P can perform an action, an instance of the probability distribution is sampled: the resulting number specifies how long it will take to complete the action in this instance.

A small but powerful set of combinators is used to build up complex behaviour from simpler behaviour. The combinators are familiar from classical process algebra: prefix(.), choice(+), parallel composition (*cooperation*)($\underset{L}{\bowtie}$) and abstraction (*hiding*)(/). Cooperation is in fact a family of combinators since its meaning varies according to the contents of the *cooperation set* L. We use $\parallel$ to denote the special case when $L = \emptyset$ and processes are concurrent without any synchronisation.

The syntax may be formally introduced by means of the following grammar:

$$\begin{array}{rcl} S & ::= & (\alpha, r).S \mid S + S \mid C_S \\ P & ::= & P \underset{L}{\bowtie} P \mid P/L \mid C \end{array}$$

where S denotes a *sequential component* and P denotes a *model component* which executes in parallel. C stands for a constant which denotes either a sequential component or a model component as introduced by a definition. C_S stands for constants which denote sequential components. The effect of this syntactic separation between these types of constants is to constrain legal PEPA components to be cooperations of sequential processes. This is a necessary condition for the associated Markov process to be *ergodic* (amenable to steady state analysis). The formal operational semantics are shown in Figure 1.

3 Compositionality: Interaction and Independence

In this section we discuss issues of compositionality within stochastic process algebras in general, and PEPA in particular. Firstly, we consider the design choices when designing a process algebra with a stochastic interpretation. Secondly, we consider the extent to which the compositionality inherent in the model description can be exploited in the analysis of the underlying stochastic model.

3.1 Designing the language

The selection of a negative exponential distribution as the governing distribution for the action durations in PEPA and other SPA has profound consequences. In terms of the underlying stochastic process, it is the only choice which gives rise to a Markov process. In terms of the process algebra it is the only choice which preserves the well-known *expansion law* which underlies the interleaving semantics. In both cases this is due to the *memoryless* property of the exponential distribution: the time until the next event is independent of the time since the last event—the exponential distribution "forgets" how long it has already waited. Thus if we consider a process $(\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop}$, from the semantics we derive:

$$\begin{tikzcd}
& (\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop} \arrow[dl, "(\alpha, r)"'] \arrow[dr, "(\beta, s)"] & \\
\mathit{Stop} \parallel (\beta, s).\mathit{Stop} \arrow[dr, "(\beta, s)"'] & & (\alpha, r).\mathit{Stop} \parallel \mathit{Stop} \arrow[dl, "(\alpha, r)"] \\
& \mathit{Stop} \parallel \mathit{Stop} &
\end{tikzcd}$$

In a generally timed (or even deterministically timed) scenario it would be important to record the elapsed time in the intermediate states in order to know the residual time of the remaining activity. For example, the time needed to complete β in $\mathit{Stop} \parallel (\beta, s).\mathit{Stop}$ should reflect the time already taken to complete activity α. However the memoryless property of the exponential distribution tells us that the distribution of the residual time in β is the same as it was initially in state $(\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop}$ before any time had elapsed. Thus we retain the expansion law of classical process algebra:

$$\begin{array}{l} (\alpha, r).\mathit{Stop} \parallel (\beta, s).\mathit{Stop} = \\ (\alpha, r).(\beta, s).(\mathit{Stop} \parallel \mathit{Stop}) + (\beta, s).(\alpha, r).(\mathit{Stop} \parallel \mathit{Stop}) \end{array}$$

Later formalisms which incorporated general distributions either avoided the issue of residual durations by separating actions and delays (e.g. Modest [17] and IGSMP [11]), or used a finer-grained semantics such as ST-semantics to distinguish the start and stop of each action (e.g. GSMPA [10]).

3.1.1 Cooperation

Communication or parallel composition is the essence of compositionality in process algebras. It gives structure to models, indicating which actions may be undertaken concurrently, and which cannot.

The choice was made to adopt the multiway synchronisation using shared names (as in CSP) rather than complementary actions (as in CCS). This means that components or agents jointly perform actions of the same type, when the parallel composition dictates it. The motivation was to represent something more general than communication. In performance models interaction often captures resource usage and the objective of the model is to study the constraints imposed on components by competition over resources. In this context the multiway synchronisation

Prefix

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

Choice

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

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

Cooperation

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

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

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

where $R = \dfrac{r_1}{r_\alpha(E)} \dfrac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$

Hiding

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

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

Constant

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

Figure 1: PEPA Structured Operational Semantics

offered more generality. However this choice was independent of the quantification of action durations, as witnessed by the adoption of CCS-style synchronisation in the stochastic π-calculus which generates a Markov process in the same way as PEPA [48].

Nevertheless the quantification of action duration did pose a challenge for the definition of cooperation. Actions which are to be performed jointly may each have been assigned rates (durations) in their respective components. The best way to resolve what should be the rate of the shared action has been a topic of some debate. The differing solutions adopted have become the main distinguishing feature of the various SPA formalisms.

The first observation is that if we view the joint action as a "synchronisation" as in the sense of *barrier synchronisation* in parallel programming then the correct duration would be the maximum of the durations, i.e. the maximum of the random variables. The unfortunate problem is that the maximum of two or more exponentially distributed random variables is not exponentially distributed.

In PEPA it is assumed that each component has *bounded capacity* to carry out activities of any particular type, determined by the *apparent rate*. For a component P and action type α, the apparent rate of α in P, denoted $r_\alpha(P)$, is the sum of the rates of each α action enabled in P. This corresponds to the rate at which P appears to an external observer to carry out an α action, due to the superposition principle of the negative exponential distribution. The definition of cooperation in PEPA is based on the assumption that a component cannot be made to exceed its bounded capacity, meaning that the apparent rate of the shared action will be the minimum of the apparent rates of the components involved.

In TIPP the "rate" is assumed to represent work capacity in one partner of the synchronisation and work demand in the other. The rate of the shared action is then taken to be the product of the two component rates. In contrast, in EMPA it is assumed that in any synchronisation exactly one participant has an explicit representation for the rate of the activity, all other participants being *passive* with respect to this activity, prepared to proceed at the rate of the active participant. The formalisms which separate action and time evolution avoid this issue by only allowing synchronisation on untimed actions. The issue of timed synchronisation is discussed in [29] and in detail in Bradley's thesis [9].

3.1.2 Semantics and equivalence relations

The semantic rules of PEPA generate a labelled transition system, just as in the case of classical process algebra. However there are some significant differences introduced by the inclusion of quantified information. In particular it is important to note that the semantics gives rise to a *multi-transition* system i.e. it is not sufficient to record the existence of a transition or arc between two nodes. The multiplicity of the transition is important. This is because the apparent rate of a term which has two copies of the same arc, generated by two instances of the same action, will differ from that of a term with only one instance. Thus the idempotence with respect to choice is lost, i.e. $P \neq P + P$.

Once a derivation graph has been generated for a particular model this forms the basis of the Markov process on which performance analysis will be carried out. A state of the Markov process is associated with each node of the graph, and the transition rate between states is simply the sum of the rates of actions labelling arcs between the corresponding nodes. The information about action types is lost in the Markov process. It is established in [31] that this generates a unique Markov process.

At the time at which stochastic process algebras emerged little attention had been paid to *equivalence relations* for Markov processes, although there was some work dating back to the 1960s looking at *partitions* of Markov processes. Driven by the problem of state space explosion, there had been keen interest in the prospect of *aggregating* Markov processes: essentially this amounts to taking a coarser view of the state space, grouping states into clusters or lumps, and redefining the dynamics of the system at this level. Unfortunately, if this is done in an arbitrary way the resulting process does not retain the Markov property. In 1960 Kemeny and Snell [40] established that if the partition of the state space has a property called *lumpability* then the resulting aggregated system maintains the Markov property. A consequence of this is that the aggregated system can be solved to yield *exact* results with respect to the original Markov process.

Definition 3.1 (Kemeny and Snell [40]) χ *is a* strongly lumpable *partition of Markov process $X(t)$ with state space $\mathcal{S} = \{s_0, \ldots, s_N\}$ if for any $X_{[l]}, X_{[k]} \in \chi$, $s_i, s_j \in X_{[k]}$*

$$q(s_i, X_{[l]}) = q(s_j, X_{[l]})$$

where $q(s, X)$ denotes the aggregated transition rate from state s to partition X.

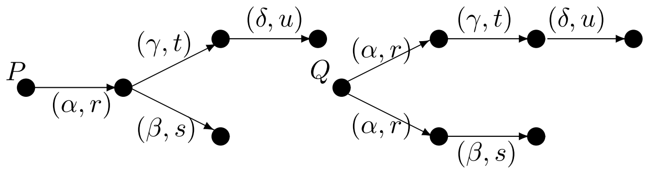
PEPA has been equipped with a number of equivalence relations which have been shown to be useful for a variety of purposes [31]. The most significant is *strong equivalence*, sometimes termed *Markovian bisimulation*. Just as with the bisimulation for classical process algebra, the central notion here is that each of a pair of components should be able to mimic the behaviour of the other from the perspective of an external observer. This observer is now assumed to have the ability to time the behaviour over many repetitions and thus deduce information about the apparent rates of actions. This means that for components to be strongly equivalent they must have the same apparent rate for all action types.

Definition 3.2 (Strong equivalence) *An equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a* strong equivalence *if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $[C] \in \mathcal{C}/\mathcal{R}$*

$$q(P, [C], \alpha) = q(Q, [C], \alpha)$$

where $q(P, [C], \alpha)$ is the total conditional transition rate, i.e. the total rate of transition from the state P to the equivalence class of states $[C]$ via α type activities.

Consider the two components P and Q, shown below:



Disregarding the activity rates they are equivalent with respect to traces but distinguished by bisimulation because when Q has performed an α action it cannot offer the choice of β and γ in the manner of P. In the Markovian setting we can immediately deduce they are not bisimilar because the apparent rates with respect to α in the initial states are not the same. P has apparent rate r for α while Q has apparent rate $2r$. Note that this is a bisimulation in the same style as the bisimulation defined by Larsen and Skou for a probabilistic variant of CCS [44].

An important property of strong equivalence in PEPA is that it induces a lumpable partition on the underlying Markov process. Thus when the PEPA process is considered only up to bisimulation and a state in the stochastic process is associated with each equivalence class of states rather than with individual states, the resulting stochastic process still has the Markov property. This forms the basis of an exact model reduction technique, termed *aggregation*. Moreover, due to the congruence property of strong equivalence the reduction technique can be applied compositionally within a model [30].

3.2 Independence

In common with most state-based modelling techniques, SPA models suffer from problems of *state space explosion* — in many instances the matrix characterising the underlying Markov process, and the corresponding steady state probability vector, are simply too large to be readily stored on standard computing equipment. A variety of approaches to this problem for numerical solution of Markov processes have appeared in the literature, including using disk-based storage [18], Kronecker and BDD-based representation of these entities [33, 26] and decomposed solutions of various forms [32].

Much work on PEPA has studied the extent to which the compositional structure in the process algebra description can be exploited in the solution of the underlying Markov process, i.e. if the Markov processes corresponding to the components can be solved separately and their solutions combined to obtain a solution of the whole Markov process. This work is discussed in this subsection.

3.2.1 Product form models

One class of Markov processes which are susceptible to an efficient solution technique are those which exhibit a *prod-*

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

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 in performance analysis. For example, Jackson networks [38] and their generalisation BCMP-networks [5], have been widely employed. In these cases the underlying Markov process is known to have a *reversible* or *quasi-reversible* structure.

Work on finding PEPA models which give rise to product form solutions has drawn on the previous work on queueing networks. Essentially this can be seen as an investigation of when components *interact* and yet remain *statistically independent* at steady state. It is clear that when a PEPA model consists of completely independent components, i.e. $P \parallel Q$, the steady state distribution will have product form:

$$\pi(P \parallel Q) = \pi_P(P) \times \pi_Q(Q)$$

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. The challenge has been to find circumstances in which interacting components P and Q exhibit statistical independence. A number of classes of such models have been identified and these fall broadly into two families. In the first family we introduce a limited form of direct interaction between components (*reversible* and *quasi-reversible* models). Here the activities on which components cooperate and the role that these activities play within the components may be restricted. In the second family there is no direct interaction between components, but they indirectly interact via contention for another component (*Boucherie resource contention* and *queueing discipline* models). We discuss the essence of each class and the resulting product form in the following paragraphs.

Reversible models 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. More formally, an irreducible, stationary Markov process $X(t)$ is *reversible* if it satisfies the detailed balance equations:

$$\pi(j)q(j,k) = \pi(k)q(k,j) \tag{3.1}$$

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 the structure of the state space of SPA models giving rise to reversible Markov processes was presented by Bhabuta *et al.* in [7]. In [34], Hillston and Thomas identified syntactic conditions which a PEPA model must satisfy in order for the underlying process to be reversible. The problem is tackled in two stages. First, a basic class of sequential components which give rise to reversible structures are identified. Then, assuming that a known class of reversible PEPA components exist, the authors investigate under what circumstances the conditions for reversibility will be preserved if reversible components are composed using the combinators of the PEPA language.

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 leads back to the original state. This ability to "undo" any transition in the subsequent transition seems to be fundamental to reversibility. It is clear that this is a necessary condition for equation 3.1 to be satisfied.

Quasi-reversible models Like reversibility, *quasi-reversibility* originates in queueing theory. 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*:

$$\pi(i) \sum_{j \in S'} q(i,j) = \sum_{j \in S'} \pi(j) q(j,i) \tag{3.2}$$

for all states i and a corresponding subset of states S'. More details of the definition of quasi-reversibility can be found in [39].

In [23], a PEPA characterisation of this class is presented. As in the work on reversibility, the approach is to first find simple instances of PEPA processes which give rise to quasi-reversible structure in their associated Markov processes. Then, conditions are established under which these components can be composed whilst maintaining the quasi-reversible property. Again the notion of a *reverse pair* is important and strong restrictions are placed on the interactions between components: each must be a *flow cooperation*. This means that the "positive" half of a reverse pair in one component is carried out in cooperation with the "negative" half of a reverse pair in another.

Boucherie resource contention models Other classes of models have been considered in which the interaction between components is indirect i.e. the components themselves are composed in parallel (without cooperation) in the model definition, but they compete over cooperation with a third component. Boucherie characterised a class of Markov processes which fit into this framework. In his definition, otherwise independent Markov processes compete for exclusive access to shared resources, causing blocking while a resource is held [8].

In [35] Hillston and Thomas characterise this class of Markov processes in PEPA. As in the underlying Markov models, the PEPA models consist of non-interacting components which give rise to the constituent processes of the Markov process. These components compete, via synchronisation with *resource components*. A PEPA component is termed a *resource* if it is never free to act independently. The general form of these process algebra terms and the resulting product form is, schematically:

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

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 cooperation with the resource (R) in isolation. Although presented here informally, these conditions are defined as formal syntactical conditions which can be checked on the model specification.

Queueing discipline models In his PhD thesis [13], Clark defined a new combinator $Q_{A,\xi}$ for PEPA which forces sequential components within its scope to observe *first-come-first-served* (FCFS) discipline with respect to action types within the set A. Moreover the rates of activities of those types are no longer controlled by the individual components but by the vector ξ. This is a *derived* combinator, meaning that any expression involving the combinator can be re-expressed using the existing PEPA combinators. In particular for a set of components, $S_1, \ldots, S_n$,

$$Q_{A,\xi}(S_1, \ldots, S_n) \equiv (S_1 \parallel \cdots \parallel S_n) \underset{M_\xi}{\bowtie} R_\xi$$

for suitably chosen M_ξ and R_ξ.

This class of models is shown to be *insensitive* and therefore to have a product form solution so that the steady state probability of the complete model can be written as an expression involving the steady state probabilities of the individual models solved in isolation. In [15] it is established that this class of models is related to BCMP queueing networks, capturing infinite server and FCFS stations *from the user's perspective*.

Unlike the other classes which have been discussed above, characterisation does not necessitate the definition of syntactic rules which may be used to check whether any model instance belongs to the class of not. Instead the use of the derived combinator means that models can be constructed with a guaranteed product form solution.

The advantage of characterising these classes of models in terms of PEPA 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.

Recent work on product form PEPA models has taken a slightly different form. In Harrison's work on the *Reversed Compound Agent Theorem* (RCAT) the process algebra has been used to establish a framework in which the relationships between different classes of product form Markov processes can be compared [22]. Within this framework Harrison has been able to demonstrate that the product forms which arise in Jackson networks [38] and G-networks [20] are based on the same fundamental mechanisms: this becomes apparent when they are represented in PEPA.

4 Enhanced qualitative evaluation

It quickly became apparent that SPA offered an excellent framework in which both qualitative and quantitative aspects of a system could be captured. However it also became apparent that while the use of logics and other formal tools for querying the functional behaviour of process algebra models was well-developed there was no equivalent formal apparatus for performance analysis. When a Markov process is analysed, either for steady state or transient behaviour, the result is a probability distribution over the entire state space. This is rarely, if ever, the final objective of modelling. Yet little attention had been given to developing formal techniques and tools for querying performance models. The advent of SPA provided some of the impetus for such work to begin.

At the most basic level the modeller wishes to construct a *reward structure* over the state space of the Markov process, to be used in conjunction with the probability distribution vector to derive performance measures. For steady state measures the reward structure is itself a vector recording a "reward" for each state, although for many states the reward value will be zero. Thus the problem becomes one of identifying the appropriate set of states to attach a non-zero reward to. Clearly, when the Markov process arises from a SPA model we prefer to characterise the states of interest at the process algebra level.

Within the context of PEPA Clark *et al.* developed a stochastic logic, PML$_\mu$, for this purpose [14]. Inspired by the probabilistic modal logic of Larsen and Skou, PML [44], PML$_\mu$ is able to differentiate PEPA terms which perform the same activities but at different rates. The key to this is a modification to Hennessy-Milner logic in which the diamond operator becomes decorated with a rate. The semantics of an expression in the logic is a subset of states, and thus logical expressions may be used, in conjunction with a value, to specify a reward structure.

In an alternative approach Argent-Katwala *et al.* have recently developed *stochastic probes* to express soft performance bounds [2]. These probes are mapped into PEPA terms and then considered in cooperation with the original PEPA model in order to extract particular performance measures in both steady state and transient behaviour.

Both PML$_\mu$ and stochastic probes seek to provide a formal framework for the calculation or derivation of performance measures. A more standard model checking approach is represented by the work on Continuous Stochastic Logic (CSL) [3]. Introduced by Aziz *et al.* in 1996 CSL is a modal logic which may be used to express temporal properties over continuous time Markov chains (Markov processes). Influenced by performance analysis, it was later extended by Baier *et al.* to incorporate a steady state operator [4]. Performance properties as well as integrated qualitative and quantitative measures (*performability* measures) are readily expressed in CSL [24]. However the approach is the usual model-checking one — a formula is checked against a model and returns *true* or *false*[1] There is no support for deriving the actual measure indicated by the model.

PEPA models have access to CSL model checking through the PRISM tool [43]. PRISM is a probabilistic model checker developed by Kwiatkowska's group at the University of Birmingham. It supports discrete time Markov chains and Markov decision processes as well as Markov processes. The standard input to PRISM is a model described in a simple reactive modules language. PEPA was integrated into the tool via a compiler which translates PEPA models into this language.

5 Accessibility

One of the beliefs of the originators of stochastic process algebras was that the formalisms would make performance evaluation accessible to a wider audience. Whilst this has perhaps been the case within academia, with other academics undertaking performance analysis which they might not have otherwise undertaken (e.g. [19, 36]) it is not true more generally. The expectation was perhaps somewhat naïve. For although it is true that languages such as LOTOS have been adopted within an industrial context it is nevertheless within a highly specialised one.

In order to really increase the accessibility of performance analysis techniques amongst the software engineering community we have to be realistic about the formalisms software designers are prepared to use. In the CEC-funded DEGAS project[2] we have sought to take advantage of the popularity of the Unified Modelling Language (UML). Within the project we have developed a framework in which suitably annotated UML models may be used to assess performance and security properties of software designs, via process algebra models [1]. PEPA is used for performance analysis whilst LySa, a variant of the Spi calculus, is used for security analysis [45].

Formal mappings from UML diagrams to process algebra models have been developed and implemented. The key functionality is provided by a pair of software modules, the *extractor* and the *reflector* [12]. For performance analysis these form a bridge between the UML modelling environment and the PEPA tools. Annotations are added to the UML model according to a pre-determined stereotype. The UML is then saved in the usual way in XMI format. The extractor produces a corresponding PEPA format which can be loaded into the PEPA Workbench. This allows a steady state probability distribution corresponding to the states of the PEPA model to be derived. However this is still inaccessible to the UML modeller — it is essential that results are reported in terms which make sense to the software designer, i.e. in terms of the original UML model. This functionality is provided by the reflector module which aggregates the steady state probability distribution data to produce suitable annotations to the UML model.

This scheme is not specific to UML models. For example, an extractor has recently been developed for models developed in BPEL4WS, a web service composition language [47].

6 Conclusions

Stochastic process algebra has gained acceptance as one of the suite of techniques available for performance modelling. Moreover, these formalisms have opened a door between theoretical computer science and the performance community, creating new opportunities for interaction and new directions for research.

The extent to which SPAs have been able to address the problems which originally motivated them is varied. Perhaps the most success has been gained in the area of integrated qualitative and quantitative evaluation of systems. The issue of accessibility is now being tackled, albeit

[1]N.B. Counter-examples are not given in the case of a *false* judgement due to the quantified nature of the formulae c.f. [42].

[2]*Design Environments for Global ApplicationS* project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing

through the intermediary of the UML. Decomposed solution of Markov processes remains a difficult topic but considering it from the process algebra perspective is providing new insight as well as practical techniques.

Meanwhile the formalisms are attracting new users from wider domains of application. Both the stochastic π-calculus and PEPA have recently been used for modelling and evaluating biochemical signalling pathways. These new domains bring new challenges for both expression and evaluation.

References

[1] DEGAS project web pages. `http://www.omnys.it/degas/`, 2005.

[2] A. Argent-Katwala, J. Bradley, and N. Dingle. Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. In *Proc. of 4th Int. Workshop on Software and Performance*, pages 49–58, Redwood Shores, California, USA, Jan. 2004. ACM Press.

[3] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous-time Markov chains. In *Proc. of 8th Int. Conference on Computer Aided Verification (CAV)*, volume 1102 of *LNCS*. Springer Verlag, 1996.

[4] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *International Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 146–161. Springer-Verlag, 1999.

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

[6] M. Bernardo and R. Gorrieri. A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. *TCS*, 202:1–54, 1998.

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

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

[9] J. Bradley. *Towards Reliable Modelling with Stochastic Process Algebras*. PhD thesis, Department of Computer Science, University of Bristol, 1999.

[10] M. Bravetti, M. Bernardo, and R. Gorrieri. From EMPA to GSMPA: Allowing for general distributions. In E. Brinksma and A. Nymeyer, editors, *Proc. of the 5th Int. Workshop on Process Algebras and Performance Modeling (PAPM '97)*, pages 17–33, 1997.

[11] M. Bravetti and R. Gorrieri. The theory of Interactive Generalized Semi-Markov Processes. *Theoretical Computer Science*, 282(1):5–32, June 2002.

[12] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, 150(2):107–120, Mar. 2003.

[13] G. Clark. *Techniques for the Construction and Analysis of Algebraic Performance Models*. PhD thesis, The University of Edinburgh, 2000.

[14] G. Clark, S. Gilmore, and J. Hillston. Specifying performance measures for PEPA. In J.-P. Katoen, editor, *Proceedings of the Fifth International AMAST Workshop on Real-Time and Probabilistic Systems*, number 1601 in LNCS, pages 211–227, Bamberg, Germany, May 1999. Springer-Verlag.

[15] G. Clark and J. Hillston. Product form solution for an insensitive stochastic process algebra structure. *Performance Evaluation*, 50(2–3):129–151, 2002.

[16] P. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.

[17] P. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. Modest — a modelling and description language for stochastic timed systems. In *Proc. of Process Algebra and Probabilistic Methods. Performance Modeling and Verification. Joint International Workshop, PAPM-PROBMIV 2001*, volume 2165 of *LNCS*. Springer-Verlag, 2001.

[18] D. Deavours and W. Sanders. An efficient disk-based tool for solving large Markov models. *Performance Evaluation*, 33:67–84, 1998.

[19] A. El-Rayes, M. Kwiatkowska, and S. Minton. Analysing performance of lift systems in PEPA. In *Proc. of 12th UK Performance Engineering Workshop*, pages 83–100, 1996.

[20] E. Gelenbe. Queueing networks with negative and positive customers. *Journal of Applied Probability*, 28:656–663, 1991.

[21] N. Götz, U. Herzog, and M. Rettelbach. TIPP—a language for timed processes and performance evaluation. Technical Report 4/92, IMMD7, University of Erlangen-Nürnberg, Germany, Nov. 1992.

[22] P. Harrison. Turning back time in Markovian process algebra. *TCS*, 290:1947–1986, 2003.

[23] P. Harrison and J. Hillston. Exploiting quasi-reversible structures in Markovian process algebra models. In *Proc. of 3rd Int. Workshop on Process Algebras and Performance Modelling*, pages 510–520. Special Issue of *The Computer Journal*, 38(7), Dec. 1995.

[24] B. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model checking performability properties. In *Proc. of Int. Conf. on Dependable Systems and Networks (DSN'02)*. IEEE Computer Society Press, 2002.

[25] H. Hermanns. *Interactive Markov Chains*. PhD thesis, IMMD7, Universität Erlangen-Nürnberg, 1998.

[26] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multiterminal binary decision diagrams to represent and analyse continuous time Markov chains. In *Proc. of 3rd Intl. Workshop on the Numerical Solution of Markov Chains*, pages 188–207, 1999.

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

[28] U. Herzog and V. Mertsiotakis. Stochastic Process Algebras Applied to Failure Modelling. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[29] J. Hillston. The nature of synchronisation. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Int. Workshop on Process Algebras and Performance Modelling*, pages 51–70, Erlangen, 1994.

[30] J. Hillston. Compositional Markovian modelling using a process algebra. In W. Stewart, editor, *Proc. of 2nd Int. Workshop on Numerical Solution of Markov Chains: Computations with Markov Chains*, Raleigh, North Carolina, Jan. 1995. Kluwer Academic Press.

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

[32] J. Hillston. *FMPA Lecture Notes*, chapter Exploiting Structure in Solution: Decomposing Composed Models. Springer-Verlag, 2001.

[33] J. Hillston and L. Kloul. An efficient Kronecker representation for PEPA models. In L. de Alfaro and S. Gilmore, editors, *Proc. of 1st PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, pages 120–135, Aachen, Germany, Sept. 2001. Springer-Verlag.

[34] J. Hillston and N. Thomas. A syntactical analysis of reversible PEPA models. In C. Priami, editor, *Proc. of 6th Workshop on Process Algebra and Performance Modelling*, pages 37–49, Nice, France, Sept. 1998.

[35] J. Hillston and N. Thomas. Product form solution for a class of PEPA models. *Performance Evaluation*, 35(3–4):171–192, 1999.

[36] D. Holton. A PEPA specification of an industrial production cell. *The Computer Journal*, 38(7):542–551, 1995.

[37] I.S.O. LOTOS : A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. IS 8807, TC97/SC21, 1989.

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

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

[40] J. Kemeny and J. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[41] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley, New York, 1975.

[42] M. Kwiatkowska. Model checking for probability and time: From theory to practice. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 351–360. IEEE Computer Society Press, 2003.

[43] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. of 12th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, number 2324 in LNCS, pages 200–204, London, UK, Apr. 2002. Springer-Verlag.

[44] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[45] F. N. M. Buchholtz, H. Riis Nielson. A calculus for control flow analysis of security protocols. *International Journal of Information Security*, 2(3–4):145–167, 2004.

[46] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[47] B. Mitchell and J. Hillston. Analysing web service composition with PEPA. In *Proc. of 3rd Workshop on Process Algebras and Stochastically Timed Activities*, pages 33–44, Edinburgh, Scotland, June 2004.

[48] C. Priami. Stochastic π-Calculus. *The Computer Journal*, 38(6), 1995.

[49] B. Strulo and P. Harrison. Spades - a process algebra for discrete event simulation. *J. Logic Computat*, 10(1):3–42, 2000.