Compositional Markovian Modelling Using A Process Algebra

J. Hillston*
Department of Computer Science,
University of Edinburgh

May 1994

Abstract

We introduce a stochastic process algebra, PEPA, as a high-level modelling paradigm for continuous time Markov chains (CTMC). Process algebras are mathematical theories which model concurrent systems by their algebra and provide apparatus for reasoning about the structure and behaviour of the model. Recent extensions of these algebras, associating random variables with actions, make the models also amenable to Markovian analysis.

A compositional structure is inherent in the PEPA language. As well as the clear advantages that this offers for model construction, we demonstrate how this compositionality may be exploited to reduce the state space of the CTMC. This leads to an exact aggregation based on lumpability. Moreover this technique, taking advantage of symmetries within the system, may be formally defined in terms of the PEPA description of the model. An equivalence relation, *strong equivalence*, developed as a process algebra bisimulation relation, is used to partition the derivation graph.

1 Introduction

The advantages of using structured modelling paradigms for Markovian modelling have been pointed out by several authors in recent years, and a number of structured approaches have been developed and exploited [1, 2, 3, 4, 5]. In this paper we present such a structured approach to CTMC construction, based on a suitably enhanced process algebra, PEPA (Performance Evaluation Process Algebra). As the name suggests, PEPA was developed to investigate a compositional approach to performance modelling. Thus, in general, a reward structure is imposed on a PEPA model which, together with the underlying CTMC, defines a Markov Reward Process (MRP) from which performance measures can be derived.

A process algebra incorporates an apparatus for reasoning about the structure and behaviour of the model. Such an apparatus is not usually available in Markovian based modelling paradigms. We present a model simplification technique for PEPA models which takes advantage of this apparatus together with the compositional nature of the language. This technique, based on exact aggregation, avoids the construction of the state space of the original model. The integrity of any reward structure defined over the model can be guaranteed. This is the major contribution of the paper. The technique is illustrated by an example.

*This work was supported by the Science and Engineering Research Council under grants 91308986 and B/93/RF/1729

2 Performance Evaluation Process Algebra

Process algebras are mathematical theories which model concurrent systems by their algebra and provide apparatus for reasoning about the structure and behaviour of the model. Examples include the Calculus of Communicating Systems (CCS) [6], and Communicating Sequential Processes (CSP) [7]. A system is characterised by its active components and the *interactions*, or *communications*, between them. Unlike queueing networks or Petri nets there is no notion of *entity* or *flow* within a model. However, in recompense, compositional reasoning is an integral part of the language.

In CCS, for example, the active components of a system are called *agents* or *processes* and these undertake *actions*, representing the discrete actions of the system. An action may be internal to an agent or may constitute the interaction or communication between neighbouring agents. Agents may proceed with their internal actions simultaneously. Combinators of the language make it possible to construct an agent which has a designated first action (prefix); has a choice over alternatives (choice); or has concurrent possibilities (composition). In PEPA prefix and choice are retained but composition is replaced by *cooperation*.

In classical process algebras, such as CCS and CSP, time is abstracted away within a process so that all actions are assumed to be instantaneous and only relative timing is represented via the traces of the process. Also all choices are nondeterministic. Timed calculi, and calculi in which nondeterministic branching is replaced by probabilistic branching via the use of a probabilistic or weighted choice, have recently been proposed [8, 9, 10, 11]. However, none of these extensions make the algebra suitable for specifying a CTMC.

Adapting the process algebra to make it suitable for Markovian modelling is achieved by associating an exponentially distributed random variable with each activity within the system. Clearly this may be regarded as an annotation of the pure process algebra model. The construction is analogous to the association of a duration with the firing of a timed transition in GSPNs and the other stochastic extensions of Petri nets.

The basic elements of the language are *components* and *activities*—these correspond to *states* and *transitions* in the underlying stochastic model. We assume that there is a countable set of possible components, $\mathcal{C}$. Each activity has an *action type* (or simply *type*). We assume that there is a countable set, $\mathcal{A}$, of all possible such types. In situations where the type of the action is unknown or unimportant it may be represented by the distinguished action type, τ, which can be regarded as the *unknown* type. Activities of this type will be private to the component in which they occur.

The duration of an activity, an exponentially distributed random variable, is represented by the parameter of the distribution. This parameter is called the *activity rate* (or simply *rate*) of the activity; it may be any positive real number, or the distinguished symbol $\top$, which should be read as *unspecified*. Thus each activity, a, is defined as a pair (α, r) where $\alpha \in \mathcal{A}$ is the action type and r is the activity rate. It follows that there is a set of activities, $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of positive real numbers together with the symbol $\top$.

PEPA is given a structured operational semantics, in the style of Plotkin [12], based on a *labelled transition system*. This provides a formal interpretation of all expressions. The underlying transition system also characterises the CTMC which represents the model. Based on the semantics notions of behavioural equivalence, or *bisimulation*, can be developed. Actions, rather than states, are considered to capture the behaviour of a system. Thus, a bisimulation is defined which equates components which are observed to perform exactly the same actions. This establishes a formal basis for the comparison and manipulation of models and components, and introduces the possibility of developing tools to automate, or semi-automate, these tasks.

2.1 Syntax and Informal Semantics

PEPA provides a small set of combinators. These combinators allow expressions, or terms, to be constructed defining the behaviour of components, via the activities they undertake and the interactions between them. The syntax for terms in PEPA is defined as follows:

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

The names of these language constructions and their interpretations are presented informally below.

Prefix: $(\alpha, r).P$ Prefix is the basic mechanism by which the behaviours of components are constructed. The component $(\alpha, r).P$ carries out activity (α, r), which has action type α and a duration which is exponentially distributed with parameter r. The component subsequently behaves as component P. When $a = (\alpha, r)$ the component $(\alpha, r).P$ may be written as $a.P$.

Choice: $P + Q$ The component $P + Q$ represents a system which may behave either as component P or as Q. $P + Q$ enables all the current activities of both P and Q. The first activity to complete distinguishes one of the components, P or Q. The other component of the choice is discarded. The continuous nature of the probability distributions ensures that the probability of P and Q both completing an activity at the same time is zero. The system will subsequently behave as P' or Q' respectively, where P' is the component which results from P completing the activity, and similarly Q'. The choice combinator represents competition between components.

Cooperation: $P \underset{L}{\bowtie} Q$ The cooperation combinator is in fact an indexed family of combinators, one for each possible set of action types, $L \subseteq \mathcal{A} \setminus \{\tau\}$. The set L, the *cooperation set*, determines the interaction between the components P and Q.

The cooperation set defines the action types on which the components must synchronise or *cooperate*. It is assumed that each component proceeds independently with any activities whose types do not occur in the cooperation set L (*individual activities*). However activities with action types in the set L require the simultaneous involvement of both components (*shared activities*). These shared activities will only be enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q. Thus one component may become blocked, waiting for the other component to be ready to participate.

In general it is assumed that both components complete some work to complete the activity cooperatively. The shared activity will have the same action type as the two contributing activities and a rate reflecting the rate of the slower participant. If an activity has an unspecified rate in a component, the component is *passive* with respect to that action type. In this case the cooperation of the component is required to achieve an activity of that type but the component does not contribute to the work involved. The unknown action type, τ, may not appear in any cooperation set.

Hiding: P/L The component P/L behaves as P except that any activities of types within the set L are *hidden*, i.e. the type of such an activity appears as the unknown type τ and the activity can be regarded as an internal delay by the component.

Hiding does not have any effect upon the activities a component may engage in individually, but it does affect whether these activities can be fully witnessed externally. Normally, when an activity is completed an external observer can see the type of the completed activity. A hidden activity is witnessed only by its delay and the unknown type, τ. Moreover such an activity cannot be carried out in cooperation with any other component. In effect the action type of a hidden activity is no longer externally accessible, to an observer or to another component. The duration of an activity is unaffected.

Constant: $A \overset{def}{=} P$ We assume that there is also a countable set of *constants*. Constants are components whose meaning is given by a defining equation such as $A \overset{def}{=} P$ which gives the constant A the behaviour of the component P. This is how we assign names to components (behaviours).

The precedence of the combinators provides a default interpretation of any expression. Hiding has highest precedence with prefix next, followed by cooperation. Choice has the lowest precedence. Brackets may be used to force an alternative parsing or simply to clarify meaning. We assume that the cooperation combinator associates to the left. However brackets may also be used to clarify the meaning of a combination of components such as $P \underset{L}{\bowtie} Q \underset{K}{\bowtie} R$, where the intended *scope* of the cooperation sets, L and K, is unclear.

The cooperation between several different components using differing cooperation sets may be regarded as being built up in layers or *levels*, each cooperation combining just two components, those components possibly being formed from cooperations between components at a lower level. This implicit layering within models will be important when we consider the model simplification technique presented in Section 4.

2.1.1 Current Action Types and Activities

The action types which the component P may next engage in are the *current action types* of P, a set denoted $\mathcal{A}(P)$.

Definition 2.1 (Set of Current Action Types)

1. $\mathcal{A}((\alpha, r).P) = \{\alpha\}$
2. $\mathcal{A}(P + Q) = \mathcal{A}(P) \cup \mathcal{A}(Q)$
3. $\mathcal{A}(P/L) = \begin{cases} \mathcal{A}(P) & \text{if } \mathcal{A}(P) \cap L = \emptyset \\ (\mathcal{A}(P) \setminus L) \cup \{\tau\} & \text{if } \mathcal{A}(P) \cap L \neq \emptyset \end{cases}$
4. $\mathcal{A}(P \underset{L}{\bowtie} Q) = \Big(\mathcal{A}(P) \setminus L\Big) \cup \Big(\mathcal{A}(Q) \setminus L\Big) \cup \Big(\mathcal{A}(P) \cap \mathcal{A}(Q) \cap L\Big)$

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

2.2 Execution Strategy

For any component P the *exit rate* from P will be the sum of the activity rates of all the activities enabled in P, i.e. $q(P) = \sum_{a \in \mathcal{A}ct(P)} r_a$. If P enables more than one activity, $|\mathcal{A}ct(P)| > 1$, then the dynamic behaviour of the model is determined by a *race condition*. This has the effect of replacing the nondeterministic branching of the pure process algebra with probabilistic branching. The probability that a particular activity completes is given by the ratio of the activity rate to the exit rate from P. Any other activities which were simultaneously enabled will be preempted. When the activities were competing (Choice), an activity which is preempted by the completion of another abandons its spent lifetime and starts another lifetime whenever it is next enabled (*preemptive restart with resampling*). In contrast when the activities are independent (Cooperation) a preempted activity will remember its spent lifetime and whenever it is next enabled it will resume from that point, only completing the remaining portion of its lifetime (*preemptive resume*).

Fortunately we can take advantage of the memoryless property of the exponential distribution. This allows a blurring of the distinction between the *preemptive restart* and the *preemptive resume* execution strategies and means that it is not necessary to retain information about the remaining lifetime of an activity in either case, as long as exponentially distributed delays are assumed for all activities.

2.3 Apparent Rate

The *apparent rate* at which an action type occurs within a component is of importance when comparing models.

Definition 2.2 *The* apparent rate *of action of type α in a component P, denoted $r_\alpha(P)$, is the sum of the rates of all activities of type α in $\mathcal{A}ct(P)$.*

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

2. $r_\alpha(P + Q) = r_\alpha(P) + r_\alpha(Q)$

3. $r_\alpha(P/L) = \begin{cases} r_\alpha(P) & \text{if } \alpha \notin L \\ 0 & \text{if } \alpha \in L \end{cases}$

4. $r_\alpha(P \underset{L}{\bowtie} Q) = \begin{cases} \min(r_\alpha(P), r_\alpha(Q)) & \text{if } \alpha \in L \\ r_\alpha(P) + r_\alpha(Q) & \text{if } \alpha \notin L \end{cases}$

Note that an apparent rate may be unspecified.

2.4 Example - Simple Resource Usage System as Cooperating Components

Consider a simple system in which a process repeatedly carries out some task. In order to complete its task the process needs access to a resource for part, but not all, of the time. Thus the task can be regarded as being in two stages: the first requiring access to the resource, the second involving only the process. The resource meanwhile is continuously available except for a short period after it has been used during which it is reset and therefore unavailable.

We model the process and the resource as two separate components: $Process$ and $Resource$ respectively. The process will undertake two activities consecutively: *use* with some rate r_1, in cooperation with the resource, and *task* at rate r_2, representing the remainder of its processing task. Similarly the resource will engage in two activities consecutively: *use*, at a rate r_3 and *update*, at rate r_4.

$$\begin{array}{ll} Process \stackrel{\mathit{def}}{=} (use, r_1).Process' & Resource \stackrel{\mathit{def}}{=} (use, r_3).Resource' \\ Process' \stackrel{\mathit{def}}{=} (task, r_2).Process & Resource' \stackrel{\mathit{def}}{=} (update, r_4).Resource \end{array}$$

$$System \stackrel{\mathit{def}}{=} Process \underset{\{use\}}{\bowtie} Resource$$

In this case it would be straightforward to model this as a single component:

$$System' \stackrel{\mathit{def}}{=} (use, r_{13}).\left((task, r_2).(update, r_4).System' + (update, r_4).(task, r_2).System'\right)$$

where $r_{13} = \min(r_1, r_3)$.

However, note that this does not as clearly reflect what is happening in the system. Moreover, representing the components of the system separately means that we can easily extend the model to represent a system in which there are two processes, independent of each other but competing for the use of the resource.

$$System2 \stackrel{\mathit{def}}{=} (Process \underset{\emptyset}{\bowtie} Process) \underset{\{use\}}{\bowtie} Resource$$

3 Operational Semantics and the Underlying CTMC

The semantics of PEPA, presented in the structured operational semantics style of [12], are shown in Figure 1. The operational rules are to be read as follows: if the transition(s) above the inference line can be inferred, then we can infer the transition below the line. The rules outline the activities which a component can witness—each completed activity brings about a transition in the system. The rules are presented without comment except for the rule defining shared activities, the third rule for cooperation.

The apparent rate of a shared action type (i.e. $\alpha \in L$) in the component $E \underset{L}{\bowtie} F$ is taken to be the slower of the apparent rates of that action type in E and F (Definition 2.2). This reflects the assumption that in general both components contribute some work on the shared activity. In the case where the apparent rate is unspecified in one component, the apparent rate will be completely determined by the other component.

Prefix

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

Choice

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

Cooperation

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

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

Hiding

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

Constant

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

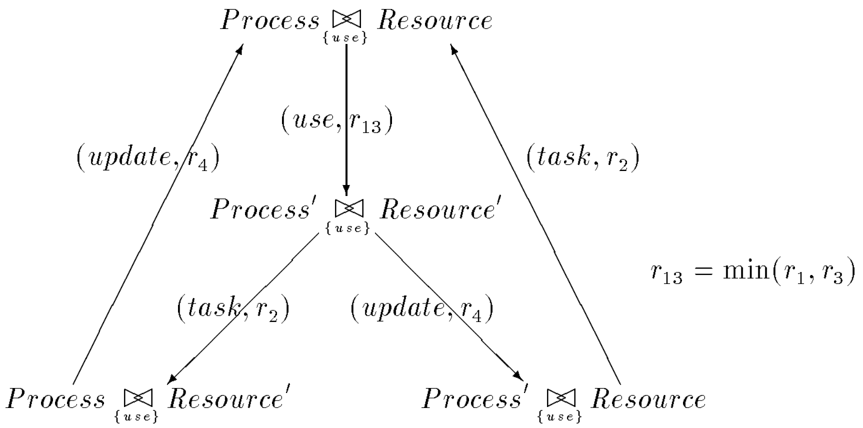
Figure 1: Operational Semantics of PEPA

We assume that the choice of an activity of the appropriate type to complete the shared activity occurs independently in the two cooperating components. For example, given that E completes an activity of type α, the probability that it is the activity $(\alpha, r_1) \in \mathcal{A}ct(E)$ is $r_1/r_\alpha(E)$. Similarly for F, given that a type α action occurs, the probability that is some instance , say (α, r_2), is $r_2/r_\alpha(F)$. Given that a shared α type activity has occurred in $E \underset{L}{\bowtie} F$ then, assuming independence of choice in E and F, the probability these two instances combined to form the shared activity is: $r_1/r_\alpha(E) \times r_2/r_\alpha(F)$. For any activity instance its activity rate is the product of the apparent rate and the probability, given that an activity of this type occurs, that it is this instance which completes. This leads to the rule for shared activities:

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

A *labelled transition system* $(S, T, \{\xrightarrow{t} \mid t \in T\})$ is a system defined by a set of states S, a set of transition labels T and a transition relation $\xrightarrow{t} \subseteq S \times S$ for each $t \in T$. PEPA can be defined as a labelled multi-transition system. In a multi-transition system a multi-relation replaces the relation because the number of instances of a transition between states is recognised. PEPA is the labelled multi-transition system $(\mathcal{C}, \mathcal{A}ct, \{\xrightarrow{(\alpha,r)} \mid (\alpha, r) \in \mathcal{A}ct\})$ where $\mathcal{C}$ is the set of components, $\mathcal{A}ct$ is the set of activities and the multi-relation $\xrightarrow{(\alpha,r)}$ is given by the rules in Figure 1.

Example - Processor/Resource System Following these rules we can construct transition diagrams representing the possible behaviours of a component. The transitions are labelled by the activities which they represent. This is often a useful representation of a component, initially more illuminating than the defining equations.



3.1 Constructing the Underlying CTMC

If we envisage a graph in which language terms form the nodes and where arcs represent the possible transitions between them, then the operational rules define the form of this graph. Since $\xrightarrow{(\alpha,r)}$ is a multi-relation, the graph we consider is a multigraph—if there is more than one instance of an arc between nodes we distinguish between them. This graph, the *derivation graph*, describing the possible behaviour of any PEPA component, provides a useful way to reason about a model. It is also the basis of the construction of the underlying CTMC.

Definition 3.1 *If* $P \xrightarrow{(\alpha,r)} P'$*, then* P' *is a* (one-step) derivative *of* P*. More generally, if* $P \xrightarrow{(\alpha_1,r_1)} \cdots \xrightarrow{(\alpha_n,r_n)} P'$*, then* P' *is a* derivative *of* P*.*

These derivatives are the states of the labelled multi-transition system. For any PEPA component the set of derivatives (behaviours) which can evolve from the component can be defined recursively.

Definition 3.2 *The* derivative set *of a PEPA component* C *is denoted* $ds(C)$ *and defined as the smallest set of components such that*

- *if* $C \stackrel{\text{def}}{=} C_0$ *then* $C_0 \in ds(C)$*;*
- *if* $C_i \in ds(C)$ *and there exists* $a \in \mathcal{A}ct(C_i)$ *such that* $C_i \xrightarrow{a} C_j$ *then* $C_j \in ds(C)$*.*

Thus the derivative set is the set of components which capture all the reachable states of the system. These form the nodes of the transition graph representing the possible behaviours of the system: the *derivation graph*.

Definition 3.3 *Given a PEPA component* C *and its derivative set* $ds(C)$*, the* derivation graph $\mathcal{D}(C)$ *is the labelled directed multigraph whose set of nodes is* $ds(C)$ *and whose multiset of arcs* A *is defined as follows:*

- *The elements of* A *are taken from the set* $ds(C) \times ds(C) \times \mathcal{A}ct$*;*
- $\langle C_i, C_j, a \rangle$ *occurs in* A *with the same multiplicity as the number of distinct inference trees which infer* $C_i \xrightarrow{a} C_j$*.*

The initial component C_0*, where* $C \stackrel{\text{def}}{=} C_0$*, is taken to be the initial node of the graph.*

The derivation graph has the component defining the model as its initial node. Each subsequent derivative is a node in the graph and there is an arc between nodes, labelled by the action type and the activity rate, for each possible transition between the corresponding components.

To form the underlying CTMC a state is associated with each node of the graph, and the transitions between states are derived from the arcs of the graph. The total transition rate between two states will be the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph. This use of the derivation graph is analogous to the use of the reachability graph in stochastic extensions of Petri nets such as GSPNs [13]. We assume that the model is finite so that the number of nodes in the derivation graph is finite.

Theorem 3.1 *For any finite PEPA model* $C \stackrel{\text{def}}{=} C_0$*, if we define the stochastic process* $X(t)$*, such that* $X(t) = C_i$ *indicates that the system behaves as component* C_i *at time* t*, then* $X(t)$ *is a continuous time Markov chain.*

The *transition rate* between two components C_i and C_j is denoted by $q(C_i, C_j)$. This is the rate at which the system changes from behaving as component C_i to behaving as C_j. It will be the sum of the activity rates labelling arcs which connect the node corresponding to C_i to the node corresponding to C_j in the derivation graph, i.e.

$$q(C_i, C_j) = \sum_{a \in \mathcal{A}ct(C_i|C_j)} r_a$$

where $\mathcal{A}ct(C_i|C_j) = \{\!|a \in \mathcal{A}ct(C_i) \mid C_i \xrightarrow{a} C_j|\!\}$. Typically this multiset will only contain one element. Clearly if C_j is not a one-step derivative of C_i, $q(C_i, C_j) = 0$.

The $q(C_i, C_j)$, or q_{ij}, are the off-diagonal elements of the infinitesimal generator matrix of the Markov process, **Q**. Diagonal elements are formed as the negative sum of the non-diagonal elements of each row, i.e. $q_{ii} = -q(C_i)$. The PEPA Workbench [14] is a suite of software tools which may be used to analyse PEPA models. In particular it can generate the generator matrix corresponding to a model in a form suitable for solution by any standard numerical technique.

We assume that all PEPA models are time homogeneous since the rate and type of an activity are independent of time, as are the activities available within a component. The ergodicity of the CTMC can be readily checked by conditions on the structure of the process term representing the model [15].

The *conditional transition rate* from C_i to C_j via an action type α is denoted $q(C_i, C_j, \alpha)$. This is the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph which are also labelled by the action type α. It is the rate at which a system behaving as component C_i evolves to behaving as component C_j as the result of completing a type α activity.

If we consider a *set* of possible derivatives S, the *total conditional transition rate* from C_i to S, denoted $q[C_i, S, \alpha]$, is defined to be

$$q[C_i, S, \alpha] = \sum_{C_j \in S} q(C_i, C_j, \alpha)$$

3.2 CTMC to MRP: Reward Structures

Information about the *types* of activities, which is contained in the derivation graph, is not incorporated into the CTMC representation. However this information may be used to define a *reward structure* over the CTMC, resulting in a Markov Reward Process (MRP).

In the framework for reward structures introduced by Howard [16], rewards are associated with states of a CTMC or with transitions between states. Rewards which accumulate continuously while a process is resident within a state are termed *yield functions*. The discrete rewards made when the process changes states are termed *bonuses*. Befitting the process algebra definition of behaviour in terms of activities, rather than states, we define a reward structure in terms of yield functions based on activities of a PEPA model. The reward associated with a component, and the corresponding state, is then the sum of the rewards attached to the activities it enables. If ρ_i is the reward associated with component C_i and $\rho(a)$ is the reward associated with the activity a then $\rho_i = \sum_{a \in \mathcal{A}ct(C_i)} \rho(a)$. In this way, the rewards are defined at the level of the PEPA model, rather than at the level of the underlying CTMC. This is similar to the use of reward structures with stochastic Petri nets, e.g. stochastic Reward nets [17], GSPN reward models [18] and stochastic activity networks (SANs) [2].

A PEPA model together with a reward structure specifies a MRP. Different reward structures may be constructed over a model, meaning that a model can be adapted to different purposes. The compositional nature of the modelling paradigm introduces the possibility of models constructed from a library of existing components. The PEPA Workbench will derive the underlying CTMC and the related MRP automatically: the modeller does not need to attempt these complex and error-prone tasks manually. Moreover the clear structure inherent in the process algebra allows the modelling task to be approached systematically.

However the author believes that the greatest attraction of the approach lies in the structure it provides within the constructed model. In the rest of the paper we will describe how this compositional structure within models may be exploited, in conjunction with a formally defined model simplification technique, resulting in an exact aggregation which avoids the need to construct the generator matrix of the original model.

4 Strong Equivalence and Exact Aggregation

In a process algebra, actions, rather than states, are taken to capture the behaviour of a system or model. This leads to a formally defined notion of equivalence, bisimulation, in which components are regarded as equal if, under observation, they appear to perform exactly the same actions. For PEPA a bisimulation called *strong equivalence* can be developed at the process algebra level and exploited by the underlying CTMC [15]. To the best of the author's knowledge this work represents the first application of the bisimulation approach to equivalence to Markov chains.

Strong equivalence is developed in the style of Larsen and Skou's probabilistic bisimulation [11]: in PEPA transition rates are used analogously to probabilities in their process algebra. It is assumed that observation occurs without detailed knowledge of the individual activities involved. For example, strong equivalence is unable to distinguish between the two components:

$$(\alpha, r).(P+Q) + (\alpha, r).(Q+P) \qquad \text{and} \qquad (\alpha, 2r).(P+Q).$$

Probabilistic bisimulation forms equivalence classes over the set of agents such that, for any two agents within a class, the probability of performing a given action, α, resulting in one-step derivatives which lie within a given equivalence class, is the same. To define strong equivalence over PEPA components we consider the *conditional transition rates* rather than the conditional transition probabilities. Two PEPA components are strongly equivalent if there is an equivalence relation between them such that, for any action type α, the total conditional transition rates from those components to any equivalence class, via activities of this type, are the same.

Definition 4.1 *An equivalence relation over components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a* strong equivalence *if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$,*

$$q[P, S, \alpha] = q[Q, S, \alpha]$$

It is clear that the identity relation trivially satisfies the Definition 4.1, and so all components are members of some strong equivalence. The following proposition ensures that any union of strong equivalence relations is itself a strong equivalence.

Proposition 4.1 *Let each $\mathcal{R}_i$, $i \in I$ for some index set I, be a strong equivalence. Then the transitive closure of their union, $\mathcal{R} = (\bigcup_{i \in I} \mathcal{R}_i)^*$, is also a strong equivalence.*

Based on this we can define the maximal strong equivalence as the union of all strong equivalences.

Definition 4.2 *We say P and Q are* strongly equivalent*, written $P \cong Q$, if $(P, Q) \in \mathcal{R}$ for some strong equivalence $\mathcal{R}$, i.e.*

$$\cong \; = \; \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a strong equivalence}\}$$

Using standard process algebra techniques we can establish that the relation $\cong$ is a congruence for the language PEPA [15]. In particular we show that the relation is preserved by the combinators of the language:

Proposition 4.2 (Preservation by Combinators)
If $P_1 \cong P_2$ then

1. $a.P_1 \cong a.P_2$;
2. $P_1 + Q \cong P_2 + Q$;
3. $P_1 \underset{L}{\bowtie} Q \cong P_2 \underset{L}{\bowtie} Q$;
4. $P_1/L \cong P_2/L$.

If we can find a model simplification technique which respects the strong equivalence relation, this implies that part of a model, represented by a component, P_1 say, can be simplified in isolation. If P_2 is strongly equivalent to P_1, but simpler in terms of the derivative set, P_2 may be used to replace P_1 without jeopardising the integrity of the whole model: the two components appear the same from the point of view of other components with which they might interact.

4.1 Strong Equivalence and the Underlying CTMC

Each equivalence class $S \in \mathcal{C}/\cong$ is a set of components all exhibiting the same pattern of behaviour. From the definition of $\cong$ it is clear that if $P \cong Q$ then the action sets of the two components are equal, i.e. $\mathcal{A}(P) = \mathcal{A}(Q)$. Moreover, since the equivalence classes $S \in \mathcal{C}/\cong$ partition the set $\mathcal{C}$, it follows that the apparent rates and the exit rates of the two components are the same:

$$r_\alpha(P) \; = \; r_\alpha(Q) \quad \text{for all } \alpha \in \mathcal{A}, \qquad \text{and} \quad q(P) \; = \; q(Q).$$

Similarly the conditional probability of completing an activity of type α and resulting in a derivative within a given equivalence class, S, is the same in the two components.

Although it is developed naturally in the process algebra, the strong equivalence relation is closely related to the notion of lumpability in a CTMC [19].

Definition 4.3 *A CTMC, $\{X_i\}$, is* lumpable *with respect to some partition $\chi = \{X_{[k]}\}$, if, and only if, for any $X_{[k]}, X_{[l]} \in \chi$, and $X_i, X_j \in X_{[k]}$,*

$$q(X_i, X_{[l]}) = q(X_j, X_{[l]})$$

where $q(X_i, X_{[l]})$ is the aggregated transition rate from X_i to all states in $X_{[l]}$, $\sum_{X_m \in X_{[l]}} q(X_i, X_m)$.

Since the relation $\cong$ partitions the set of components $\mathcal{C}$, if it is restricted to the derivative set of any component P, the relation partitions this set. Let $ds(P)/\cong$ denote the set of equivalence classes generated in this way.

Proposition 4.3 *For any component P, $ds(P)/\cong$ induces a lumpable partition on the state space of the CTMC corresponding to P.*

Proof Let S_i and S_j denote arbitrary elements of $ds(P)/\cong$, and consider any two elements of S_i, P_{i_k} and P_{i_ℓ}. Then since $P_{i_k} \cong P_{i_\ell}$,

$$q[P_{i_k}, S_j] \; = \; q[P_{i_\ell}, S_j]$$

Thus, it follows immediately that the partition $ds(P)/\cong$ induces a lumpable partition on the state space of the CTMC underlying P. $\square$

4.2 Strong Equivalence for Aggregation

The partition formed on the derivative set of any PEPA model by strong equivalence may be used to provide exact aggregation of the underlying CTMC. It follows immediately from Proposition 4.3, and the definition of lumpability [19], that if strong equivalence is used to induce a partition of the state space of the CTMC then the aggregation will result in a CTMC. Thus the aggregated process may be solved exactly to find the steady state distribution. The reduced state space is formed by associating a state with each equivalence class induced on the derivative set, $S \in ds(P)/\cong$. In most cases this new process will have a smaller state space than the original model.

For any PEPA component $\mathcal{S}$, let the set of equivalence classes, $ds(\mathcal{S})/\cong$, be called the *lumped derivative set*. For any element, T, of this set we can construct the *lumped activity set*, $\mathcal{A}ct_{\cong}(T)$.

Definition 4.4 *Suppose that P is an arbitrary element of $T \in ds(\mathcal{S})/\cong$, then the* lumped activity set *of T, $\mathcal{A}ct_{\cong}(T)$ is*

$$\mathcal{A}ct_{\cong}(T) = \{(\alpha, q') \mid r_\alpha(P) \neq 0, q' = q[P, S, \alpha] \textit{ for some } S \in ds(\mathcal{S})/\cong\}$$

Moreover the complete lumped activity set *of the component $\mathcal{S}$ is,*

$$\vec{\mathcal{A}ct}_{\cong}(\mathcal{S}) = \bigcup_{T \in ds(\mathcal{S})/\cong} \mathcal{A}ct_{\cong}(T)$$

Based on this we can also define the *lumped derivation graph*.

Definition 4.5 *Given a PEPA component $\mathcal{S}$, and its lumped derivative set $ds(\mathcal{S})/\cong$, the* lumped derivation graph, *$\mathcal{D}_{\cong}(\mathcal{S})$, is the labelled directed graph whose set of nodes is $ds(\mathcal{S})/\cong$ and whose set of arcs, $A_{\cong}$ is defined as follows:*

- *The elements of $A_{\cong}$ are taken from the set $ds(\mathcal{S})/\cong \times ds(\mathcal{S})/\cong \times \vec{\mathcal{A}ct}_{\cong}(\mathcal{S})$;*
- *$\langle T_i, T_j, (\alpha, q')\rangle \in A_{\cong}$ if $(\alpha, q') \in \mathcal{A}ct_{\cong}(T_i)$ and $q' = q[P_i, T_j, \alpha]$ for all $P_i \in T_i$.*

The node T_0, where $\mathcal{S} \in T_0$, is taken to be the initial node of the graph.

The new (aggregated) CTMC is derived from the lumped derivation graph analogously to the way in which the original CTMC was derived from the derivation graph of the complete model.

If the integrity of any performance measure to be extracted from the model is to be maintained we must also consider the effect of the aggregation on the MRP. In [20] Nicola extends the notion of strong lumpability to MRP.

Definition 4.6 *A MRP is* strongly lumpable with respect to a reward R *in the context of a partition* χ*, if, for every initial distribution, the aggregated process is a MRP which results in the same reward.*

In general, the lumped reward for any element $T_j \in ds(\mathcal{S})/\cong$ is defined in terms of the conditional steady state probabilities for each component within the equivalence class. If $\hat{R}$ denotes the lumped reward, corresponding to the reward R, and Π_j is the conditional steady state probability,

$$\hat{R}(T_j) = \sum_{C_i \in T_j} R(C_i)\, \Pi_j(C_i) \qquad \text{for all } T_j \in ds(\mathcal{S})/\cong$$

If all derivatives within a partition $T_j \in ds(\mathcal{S})/\cong$ attract the same reward the resulting MRP is lumpable with respect to that reward structure. However if the MRP is to be lumpable with respect to R, it must be possible to calculate the reward without disaggregation to find the Π_j. This is the case when

Proposition 4.4 *The MRP corresponding to PEPA model* $\mathcal{S}$ *and its reward structure including some reward* R*, is strongly lumpable with respect to* R*, in the context of the partition induced by* $\cong$*, if for all* $T_j \in ds(\mathcal{S})/\cong$*, for all* $C_i \in T_j$*,* $R(C_i) = \rho$*. Then* $\hat{R}(T_j) = \rho$*.*

Proof The aggregation is strongly lumpable, and results in a CTMC, by Proposition 4.3. It remains to show that the reward R, is maintained, but this follows immediately since,

$$R = \sum_{C_i \in ds(\mathcal{S})} \rho_i \Pi(C_i) = \sum_{T_j \in ds(\mathcal{S})/\cong} \left(\sum_{C_i \in T_j} \rho_i \Pi(C_i) \right) = \sum_{T_j \in ds(\mathcal{S})/\cong} \rho\, \hat{\Pi}(T_j)$$

where $\hat{\Pi}(\cdot)$ is the steady state of the aggregated process. $\square$

Since rewards are defined in terms of the activities of a component many PEPA models will satisfy the condition of Proposition 4.4. For example in order to calculate the throughput of an action within a system a reward equal to the activity rate is attached to all activities of the given action type. As the apparent rate of an action type is the same in all strongly equivalent components, this reward will satisfy the condition of Proposition 4.4.

Although it may be stated in terms of the PEPA model rather than the underlying CTMC (state space) aggregation based on strong equivalence does not offer a new approach to model simplification. Rather, it is a reformulation of the well-established approach based on lumpability. A new approach can be developed, however, if we exploit the result that strong equivalence is a congruence and hence take advantage of the compositional structure of the model. In this latter case exact aggregation can be carried out without constructing the complete derivative set, or the corresponding generator matrix, of the original model.

4.3 Compositional Aggregation

For any PEPA component, P, we can define the *minimal* component, with respect to the size of the derivative set, which is strongly equivalent to P. This is the *lumped component* of P, denoted $\hat{P}$, and $|ds(\hat{P})| \leq |ds(P)|$. If we systematically replace the components within a model by their lumped components, the derivative set of the whole model will, in general, be reduced whilst its behaviour is maintained. The implicit layering of models, based on the cooperation combinator, provides the structure for such a systematic approach. If all the components within a model are already minimal no reduction will occur, but the procedure can never increase the size of the derivative set, or the state space of the corresponding CTMC.

It is clear from the Definition 3.3 that just as a derivation graph can be formed corresponding to any PEPA component, so a PEPA component corresponding to any derivation graph can be formed. Furthermore, if we consider a lumped derivation graph, as defined in Definition 4.5, we can see that it may be regarded as a derivation graph. From the lumped derivation graph we form the lumped component.

Definition 4.7 *The* lumped component *of P, $\hat{P}$, is formed from the lumped derivation graph, $\mathcal{D}_{\cong}(P)$ in the natural way: we associate a component $\hat{P}_j$ with each $T_j \in ds(P)/\cong$ such that*

$$\mathcal{A}ct(\hat{P}_j) = \mathcal{A}ct_{\cong}(T_j)$$

In particular we associate $\hat{P}_0$ with T_0, the initial node of $\mathcal{D}_{\cong}(P)$. Then $\hat{P} \stackrel{\text{def}}{=} \hat{P}_0$.

Proposition 4.5 *Any PEPA component P is strongly equivalent to its lumped component, $\hat{P}$, i.e. $P \cong \hat{P}$.*

Proof By definition, $ds(\hat{P})/\cong \;=\; ds(\hat{P}) \;=\; ds(P)/\cong$ and the result follows immediately. □

Since strong equivalence is a congruence Proposition 4.5 implies that if we replace one component within a model by the equivalent lumped component, the new model will be strongly equivalent to the original one.

Guided by the layering within a model, and starting at the lowest level, we replace cooperating components by their strongly equivalent lumped components. This cannot increase the size of the derivative set of the model, and in most cases will reduce it, sometimes dramatically.

Proposition 4.6 *When a lumped component, $\hat{P}$, replaces a top-level component P, within a model, $\mathcal{S}$, to form a modified model, $\mathcal{S}'$, then $|ds(\mathcal{S}')| \leq |ds(\mathcal{S})|$.*

Thus we can systematically simplify a model, by considering each of its top-level components in turn. Each of these is itself treated as a separate model, and whose top-level components are identified and so on. At some level the identified top-level components will be atomic, in the sense that they cannot themselves be broken down into cooperating components. Each of these is replaced by its lumped component. At each level of the model the aggregation procedure is applied, until a lumped version of the complete original model is formed. The outline of a procedure to implement this approach to model simplification is shown in Figure 2. Note that the full derivative set of the original model does not need to be constructed. No CTMC is constructed until the aggregation procedure is complete.

1. Construct the model $\mathcal{S}$, combining components until the full functionality is represented.

2. Identify the cooperating components at the lowest level of the model, X and Y say. Apply compositional aggregation to the cooperation of these components, $P \equiv X \underset{L}{\bowtie} Y$, to form the lumped component $\hat{P}$:
   - 2.1 Form the derivative set of the component P, $ds(P)$.
   - 2.2 Find the strong equivalence classes within $ds(P)$ and form the lumped derivative set $ds(P)/\cong$.
   - 2.3 Form the lumped activity set for each $T \in ds(P)/\cong$ and construct the lumped derivation graph $\mathcal{D}_{\cong}(P)$.
   - 2.4 Construct the lumped component $\hat{P}$, based on $\mathcal{D}_{\cong}(P)$, and replace P by $\hat{P}$.

   Repeat with all other pairs of components at the lowest level of $\mathcal{S}$.

3. Consider the next level of the model, i.e. $Q \underset{K}{\bowtie} \hat{P}$, for some Q. Repeat steps $2.1 - 2.4$ applied to $Q \underset{K}{\bowtie} \hat{P}$. Continue in this way until $\hat{\mathcal{S}}$ has been formed.

4. Based on the derivation graph of $\hat{\mathcal{S}}$ (this will be the lumped derivation graph of $\mathcal{S}$ formed in the previous step) form an aggregated MRP representing the model and its reward structure.

5. Solve the aggregated CTMC and calculate the rewards.

Figure 2: Outline of a procedure implementing compositional aggregation of PEPA model $\mathcal{S}$.

As previously we assume that we restrict the lumping of components to the situations when the integrity of any performance measures can be ensured. Thus we restrict the aggregation to components where the resulting partition will be strongly lumpable with respect to R, as in the case of components which satisfy the condition of Proposition 4.4. This may mean that some components for which a lumped component exists, are left unlumped. However, it has the consequence that lumped rewards are easy to derive and no disaggregation is necessary. Moreover the approach has potential to be automated and carried out without the participation of the modeller.

4.4 Example - Asymmetric MSMQ System with Random Polling

We consider an symmetric multi-server multi-queue (MSMQ) system. The MSMQ system is an extension of the classical polling system to include more than one server. It has been shown to be a useful representation of multiple token, and multi-slotted, rings.

The system we consider has two independent servers each of which is routed randomly between the nodes. There are N nodes in the system each with a single place buffer. Buffering is assumed to be restricted, i.e. a customer occupies a place in the node until service is completed. At each node, $Node_j$, arrivals occur at rate λ and service occurs at rate μ. The PEPA model of this simple system is shown in

$$
\begin{array}{ll}
Node_{j0} \stackrel{def}{=} (in, \lambda).Node_{j1} + (pass_j, e).Node_{j0} & \text{for } 1 \leq j \leq N \\
Node_{j1} \stackrel{def}{=} (engage_j, e).Node_{j2} & \\
Node_{j2} \stackrel{def}{=} (serve, \top).Node_{j0} + (pass_j, e).Node_{j2} & \\
 & \\
S \stackrel{def}{=} \sum_{k=1}^{N} (walk, \omega/N).S_k & \\
S_k \stackrel{def}{=} (pass_k, \top).S + (engage_k, \top).(serve, \mu_k).S & 1 \leq k \leq N \\
 & \\
MSMQ \stackrel{def}{=} (Node_{10} \underset{\emptyset}{\bowtie} \cdots \underset{\emptyset}{\bowtie} Node_{N0}) \underset{L}{\bowtie} (S \underset{\emptyset}{\bowtie} S)/K & \\
\text{where } L = \{engage_j, pass_j, serve \mid 1 \leq j \leq N\}, K = \{pass_j, engage_j \mid 1 \leq j \leq N\} &
\end{array}
$$

Figure 3: PEPA model of an MSMQ System with Random Polling

$$
\begin{array}{l}
SS \stackrel{def}{=} \sum_{j=1}^{3} (walk, 2\omega/3).S_jS \\
S_jS \stackrel{def}{=} \sum_{i=1}^{3} (walk, \omega/3).S_jS_i + (pass_j, \top).SS + (engage_j, \top).S'S \\
S_jS_i \stackrel{def}{=} (pass_i, \top).S_jS + (engage_i, \top).S_jS' + (pass_j, \top).S_iS + (engage_j, \top).S_iS' \\
S_jS_j \stackrel{def}{=} (pass_j, \top).S_jS + (engage_j, \top).S_jS' \\
SS' \stackrel{def}{=} \sum_{j=1}^{3} (walk, \omega/3).S_jS' + (serve, \mu).SS \\
S_jS' \stackrel{def}{=} (serve, \mu).S_jS + (pass_j, \top).S'S + (engage_j, \top).S'S' \\
S'S' \stackrel{def}{=} (serve, 2\mu).SS'
\end{array}
$$

Figure 4: The lumped component corresponding to the servers in the case $N = 3$.

Figure 3.

Each free server *walk*s to one of the nodes, chosen at random, at rate ω. On arrival at the node it queries the node to see if there is a customer waiting to be served: this query may appear as a *pass* or an *engage* type activity depending on the result of the query. If the server passes the node it returns to the idle state; if it is engaged by the node it must first perform a *serve* activity. Note that the *walk* action of the server is represented as N possible activities in the model representing the different possible outcomes of the action. The nodes are all independent of each other but the action types $pass_j$, $engage_j$ and $serve$ can only be achieved when a node and a server cooperate.

We assume that the measure to be calculated from the model is the *customer throughput*, the mean number of customers completing service per unit time. This is derived from a reward structure in which a reward μ is associated with each activity $(serve, \mu)$. Since no reward is attributed to the action types $pass_i$ and $engage_i$ these action types are hidden at the top-level of the model.

In Figure 4 the lumped component corresponding to $S \underset{\emptyset}{\bowtie} S$, when $N = 3$ is shown. Although this component appears more complex that the compositional definition of the servers, $S \underset{\emptyset}{\bowtie} S$ shown in Figure 3 it has only 15 derivatives, rather than 25. Considered in isolation the nodes are not amenable to a similar reduction. However when we consider the complete model, and the action types $pass_i$

N	$\|ds(MSMQ)\|$	$\|ds(S \underset{\emptyset}{\bowtie} S)\|$	$\|ds(N \cdots N)\|$	$\|ds(S \underset{\emptyset}{\bowtie} S)/\cong\|$	$\|ds(MSMQ(SS))\|$	$\|ds(\overline{MSMQ})\|$
2	144	16	9	10	21	37
3	230	25	27	15	34	134
4	744	36	81	21	47	424
5	2192	49	243	28	1232	60
6	6064	64	729	36	3376	73

Table 1: Table showing the size of the derivative sets before and after aggregation.

and $engage_i$ are hidden, a further reduction can be made taking advantage of the symmetry between the nodes. In Table 1 the effect of the compositional aggregation is shown for varying values of N. Note that $ds(N \cdots N)$ denotes $ds(Nodes \underset{\emptyset}{\bowtie} \cdots \underset{\emptyset}{\bowtie} Nodes)$, $ds(MSMQ(SS))$ denotes the derivative set of the model when $S \underset{\emptyset}{\bowtie} S$ is replaced by its lumped component, and $ds(\overline{MSMQ})$ denotes the lumped component of that model.

5 Conclusions

We have presented an enhanced process algebra and explained how it may be used to specify and construct CTMC models. This represents a new approach to constructing Markovian models and a new application for process algebras. Moreover we have demonstrated that the structure contained in the model may be exploited to provide a compositional approach to aggregation. This model simplification technique based on lumpability may be applied at the process algebra level, thus avoiding the construction of the generator matrix of the original model.

References

[1] Peter Buchholz. Hierachical Markovian Models - Symmetries and Reduction. In R.J. Pooley and J. Hillston, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 10 of *EDITS*, pages 234–246. Edinburgh University Press, August 1993.

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

[3] B. Plateau, J-M. Fourneau, and K-H. Lee. PEPS: A Package for Solving Complex Markov Models of Parallel Systems. In R. Puigjaner, editor, *Proc. of 4th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 341–360. Planum Press, 1988.

[4] S. Donatelli. Superposed Generalised Stochastic Petri Nets: Definition and Efficient Solution. In M. Silva, editor, *Proc. of 15th Int. Conference on Application and Theory of Petri Nets*, 1994.

[5] K. Atif W.J. Stewart and B. Plateau. The Numerical Solution of Stochastic Automata Network. Technical Report 6, LMC-IMAG, November 1993.

[6] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[7] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[8] F. Moller and C. Tofts. A Temporal Calculus for Communicating Systems. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, volume 458 of *LNCS*, pages 401–415. Springer-Verlag, August 1989.

[9] J. Baeten and J. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[10] C-C. Jou and S.A. Smolka. Equivalences, Congruences and Complete Axiomatizations of Probabilistic Processes. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, volume 458 of *LNCS*, pages 367–383. Springer-Verlag, August 1990.

[11] K. Larsen and A. Skou. Bisimulation through Probabilistic Testing. *Information and Computation*, 94(1):1–28, September 1991.

[12] G.D. Plotkin. A Structured Approach to Operational Semantics. Technical Report DAIMI FM-19, Computer Science Department, Aarhus University, 1981.

[13] M. Ajmone Marsan, G. Conte, and G. Balbo. A Class of Generalised Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[14] 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 Technqiues and Tools for Computer Performance Evaluation*, volume 794 of *LNCS*, pages 353–368. Springer-Verlag, 1994.

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

[16] R. Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Systems*, volume II. Wiley, 1971.

[17] J.K. Muppala and K.S. Trivedi. Composite Performance and Availability Analysis Using a Hierarchy of Stochastic Reward Nets. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, pages 335– 349. Elsevier, February 1991.

[18] Gianfranco Ciardo, Jogesh Muppala, and Kishor Trivedi. On the solution of GSPN reward models. *Performance Evaluation*, 12:237–253, 1991.

[19] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[20] V. Nicola. Lumping in Markov Reward Processes. Research Report RC 14719, IBM, 1989. IBM Thomas Watson Research Centre, P.O. Box 704, Yorktown Heights, NY 10598.