Exploiting Quasi-reversible Structures in Markovian Process Algebra Models

PETER HARRISON[1] AND JANE HILLSTON[2]

[1] Department of Computing, Imperial College, 180 Queens Gate, London SW7 2BZ, UK. email: pgh@doc.ic.ac.uk
[2] Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK. email: jeh@dcs.ed.ac.uk

Efficient product form solution is one of the major attractions of queueing networks for performance modelling purposes. These models rely on a form of interaction between nodes in a network which allows them to be solved in isolation, since they behave as if independent up to normalisation.
Markovian process algebras (MPA) extend classical process algebras with information about the duration of actions but retain their compositional structure: a system is modelled as an interaction of components. The advantages of this compositional structure for model construction and model simplification have already been demonstrated.
In this paper we exploit results from queueing networks to identify a restricted form of interaction between suitable MPA components which leads to a product form solution. Each component of the model may be solved separately and the compositional structure of an MPA consequently facilitates efficient solution for successively more complex models. This work uses the notion of *quasi-reversibility* in a Markov process setting to define the type of interaction between MPA components. This leads to a substantial class of MPA definitions that have product-form solutions which is more general than the usual queueing network-based class of Markov processes.

1. INTRODUCTION

The size and complexity of many modern computer and communication systems result in large, complex performance models. Queueing networks offer a compositional approach to the construction of such models: the system is decomposed into service centres which represent separate aspects of processing, and a customer's progress through the system is represented as a journey between these nodes. Many such networks have been shown to exhibit a product form equilibrium distribution for their state space probabilities, leading to efficient solution techniques in which the service centres are solved in isolation. The efficiency of these solution techniques has contributed a great deal to the popularity of queueing networks for performance modelling.

In recent years there has been growing interest in the use of process algebra-based modelling paradigms for performance evaluation. This has led to the development of extensions to classical process algebras in which time and probabilistic elements are introduced, *stochastic process algebras* (SPA). As in a classical process algebra, a system is modelled as an interaction of *agents*; the behaviour of each agent is defined by the *actions* it can perform or as a composition of smaller agents. However, actions are no longer assumed to be instantaneous. Each has a duration characterised by a *random variable* which we take to be exponentially distributed. This yields a *Markovian process algebra* (MPA) which has a richer set of algebraic properties than a general SPA and hence is more amenable to analysis. Performance evaluation is carried out using the Markov process which may be derived from any MPA model. Several MPAs have been proposed in the literature; in this paper we use Hillston's PEPA (Performance Evaluation Process Algebra) (Hillston, 1994a) but the results may be generalised easily to other MPA languages.

The major advantage of MPAs is the clear compositional structure which is incorporated into models. This structure is formally defined and the paradigm includes apparatus such as equivalence relations which exploit it, for example to identify observationally equivalent processes at equilibrium. Unlike queueing networks, where the flow of customers necessarily *flattens* the model into a single entity, the SPA models retain their component structure. The benefits of the compositional approach for model simplification as well as model construction have already been demonstrated (Hillston, 1995). The MPA languages offer greater expressiveness than queueing networks but solution of the underlying Markov process currently relies on numerical techniques and the models are prone to problems of state space explosion.

The aim of this paper is to demonstrate that, in some cases at least, the compositional structure inherent in MPA models, may be exploited during model solution. In PEPA notation, given a model of the form

$$P \underset{L}{\bowtie} Q$$

we wish to derive syntactic conditions on P, Q and L which allow us to recognise when the equilibrium distribution of the model has a product form, i.e.

$$\Pi(P \underset{L}{\bowtie} Q) \equiv \Pi(P) \cdot \Pi(Q)$$

Quasi-reversibility, as defined in terms of queues and customers, has been used to show that queues behave as if independent (up to normalisation) in queueing networks possessing certain properties. Therefore, given information about the influence of the rest of the network on each queue, these queues may be solved in isolation. The required information is the arrival rate of customers when the system is in equilibrium and it is usually derived from traffic equations.

Quasi-reversibility imposes a strong condition on the arrival rates of customers but a weaker condition on probability flux than that imposed by *reversibility*. A reversible process must necessarily satisfy the detailed balance equations

$$\pi(x)\, q(x,x') = \pi(x')\, q(x',x) \qquad \text{for all states } x, x'$$

where q is the process's instantaneous state transition matrix. A quasi-reversible process satisfies instead *partial balance equations* of the form

$$\pi(x) \sum_{x' \in S'} q(x,x') = \sum_{x' \in S'} \pi(x') q(x',x)$$

for all states x and a corresponding subset of states S'. Thus for any state x the probability flux into and out of the subset of states S' is balanced. In queueing terms the set of states, S', comprises those states which differ from x by the arrival of exactly one customer at some server. The additional condition is that the arrival rates of customers must be independent of the current state of the queue. It has been shown that for both open and closed networks of queues, which are quasi-reversible when considered in isolation, the queues behave independently (up to normalisation in the closed case), meaning that the equilibrium distribution had a product form. Moreover the network itself forms a quasi-reversible system.

In this paper we consider how the definition of quasi-reversibility can be given, without direct reference to queues and customers, in a more general context which can be applied naturally to PEPA models. We will show that the resulting classification of quasi-reversible MPA models includes the usual queueing examples and establish when an interaction between quasi-reversible PEPA components represents an aggregate quasi-reversible system. Furthermore, we will be able to detect quasi-reversible processes outside the domain of conventional queueing networks. Interactions of the appropriate type can be readily identified syntactically and efficient solution algorithms, taking advantage of the product form, used to find the equilibrium solution.

The paper is organised as follows: in Section 2 we define quasi-reversibility in a general context and exhibit the form of interactions between quasi-reversible systems which preserve quasi-reversibility. In Section 3 the stochastic process algebra is briefly introduced and the relationship between the process algebra components and the underlying Markov process explained. Section 4 applies the definitions of Section 3 in the context of PEPA models and we identify syntactically a class of models, corresponding to queueing networks, which are quasi-reversible and so have a product form solution. This approach is illustrated by an example in Section 4.3. In Section 5 more complex examples are given which demonstated that the class of models considered is beyond the scope of conventional queueing networks. Finally in Section 6 we summarise the results of the paper and outline some directions for future work.

2. QUASI-REVERSIBILITY

Consider a stochastic process $X(t)$ the state of which can change either via an *input process* that causes one of a countable set of specified types of change in the state, say $x \to x'$, or via a dual *output process* that causes a change of state in the opposite direction, for example $x' \to x$. In queueing network theory the input process corresponds to arrivals to a queue (or subnetwork of queues) and the output process to departures.

DEFINITION 2.1. *Such a stochastic process is* quasi-reversible *if, and only if, $X(t)$ is a stationary Markov process such that for all times t_0 the state $X(t_0)$ is independent of:*

1. *the input process after t_0;*
2. *the output process before t_0.*

By Burke's theorem, the M/M/1 queue with constant rates is quasi-reversible. It is also obviously reversible, by inspection of its state transitions which form a birth-death process, but in general neither of the properties of reversibility and quasi-reversibility implies the other.

We call the thinned input process obtained by selecting only those state changes of a particular type r the *input subprocess r*. The following proposition then holds:

PROPOSITION 2.1. *For a quasi-reversible process:*

1. *the input subprocesses are Poisson with constant rate;*
2. *the output subprocesses are Poisson with constant rate.*

The proof is a direct generalisation of Kelly's (Kelly, 1979) with his "class r arrival" to a queueing system replaced by "a transition (of type r) caused by input subprocess r", and similarly for departures. The key property we require is that each input subprocess has *constant rate*, $\alpha(r)$ say for subprocess r. Using a combination of reversibility and quasi-reversibility arguments, it can be shown that a single multi-class queue and networks thereof satisfy *local* balance equations and have product-form solutions. The essence of the method is to make an informed guess at the transition rates $q'(x', x)$ characterising the *reversed* process—there are usually not many possibilities—and the product-form solution $\pi(x)$, and to verify that the following necessary and sufficient conditions hold for all states x, x':

$$\pi(x)q(x, x') = \pi(x')q'(x', x) \quad \text{and} \quad q'(x) = q(x)$$

where $q(x) = \sum_{x' \neq x} q(x, x')$ and similarly for q'.

The details of the method as applied to queues with single step transitions may be found in (Kelly, 1979) and in abridged form in (Harrison & Patel, 1992). However, here, we consider the results in more generality. For example, in a queueing context we can allow an arrival of given class r to comprise a batch of any (fixed) number b_r of tasks—not just one. In this way we can model the transmission of messages of different lengths, broken up into packet-trains, through a communication network. We return to this example in Section 4.3.

3. PEPA

The basic elements of PEPA are *components* and *activities*, corresponding to *states* and *transitions* in the underlying Markov process. Each activity has an *action type* (or simply *type*). Activities which are private to the component in which they occur are represented by the distinguished action type, τ. The duration of each activity is represented by the parameter of the associated exponential distribution: the *activity rate* (or simply *rate*) of the activity. This parameter may be any positive real number, or the distinguished symbol $\top$ (read as *unspecified*). Thus each activity, a, is a pair (α, r) where α is the action type and r is the activity rate.

3.1. Syntax and informal semantics

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

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

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

Cooperation: $\boldsymbol{P \underset{L}{\bowtie} Q}$ The components proceed independently with any activities whose types do not occur in the *cooperation set* L (*individual activities*). However, activities with action types in the set L require the simultaneous involvement of both components (*shared activities*). These activities are only enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q.

The published MPAs differ on how the rate of shared activities are defined (Hillston, 1994b). In PEPA the shared activity occurs at the rate of the slowest participant. If an activity has an unspecified rate in a component, the component is *passive* with respect to that action type. This means that the component does not influence the rate at which any shared activity occurs. The cooperation combinator associates to the left but brackets may also be used to clarify the meaning.

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

Constant: $\boldsymbol{A \stackrel{def}{=} P}$ Constants are components whose meaning is given by a defining equation; $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P. This is how we assign names to components (behaviours). There is no explicit recursion operator but components of infinite behaviour may be readily described using sets of mutually recursive defining equations. An example of this—the simple single server queue—is used in Section 4.

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

3.2. Execution strategy

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

3.3. Operational semantics and the underlying CTMC

The semantics of PEPA, presented in the structured operational semantics style, is given in (Hillston, 1994a). The underlying transition system also characterises the Markov process represented by the model. 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 semantic rules. The states of the labelled multi-transition system are *derivatives*. If $P \xrightarrow{(\alpha,r)} P'$, then we say that P' is an (α, r)-*derivative*, or an α-*derivative*, of P. The *derivative set*, denoted $ds(P)$, of a component P is defined recursively (see (Hillston, 1994a) for details).

The *derivation graph* is a graph in which syntactic terms form the nodes, and arcs represent the possible transitions between them: the operational rules define the form of this graph. Since $\xrightarrow{(\alpha,r)}$ is a multi-relation, the graph is a multigraph.

This derivation graph describes the possible behaviour of any PEPA component and provides a useful way to reason about a model. It is also the basis of the construction of the underlying Markov process: a state is associated with each node of the derivation graph, and the transitions between states are derived from the arcs of the graph. The *transition rate* between two components C_i and C_j, denoted $q(C_i, C_j)$, is the sum of the activity rates labelling arcs connecting node C_i to node C_j. This use of the derivation graph is analogous to the use of the reachability graph in stochastic extensions of Petri nets such as GSPN (Ajmone Marsan *et al.*, 1984). In previous work on PEPA it has been assumed that models were finite so that the number of nodes in the derivation graph was finite. This was because solution was carried out numerically, by solving the global balance equations. However, here we consider infinite state models.

Recall that the "states" of a PEPA model as it evolves are the syntactic terms, or derivatives, which the model will go through. When a model is defined to consist of one or more cooperating components, these will be apparent in every derivative of the model*. They, and the cooperation sets in operation between them, will remain static throughout the evolution of the model. Only the particular derivatives exhibited by each of the cooperating components may change. This suggests a more compact representation of the states a model may pass through.

DEFINITION 3.1. *Let P be a PEPA model comprising cooperating components $C_1, C_2, \ldots C_K$. Then a* state vector *of the model component P at derivative P_i is the vector $(C_{1_i}, C_{2_i}, \ldots, C_{K_i})$ where $C_{k_i}, 1 \leq k \leq K$ is the current derivative of C_k in P_i.*

4. QUASI-REVERSIBLE STRUCTURES IN PEPA MODELS

We now consider the problem of identifying cases when the CTMC underlying a PEPA model has a product form equilibrium distribution which corresponds to the cooperating components within the model. In these cases the probability of a given model derivative will be the product of the probabilities of the corresponding derivatives in the cooperating components, possibly subject to renormalisation. In other words, if the derivative P_i is expressed in state vector representation as $(C_{1_i}, C_{2_i}, \ldots, C_{n_i})$, then Π, the steady state probability distribution over the derivatives of P, can be expressed as

$$\Pi(P_i) = \frac{1}{G} \prod_{j=1}^{n} \pi_j(C_{j_i})$$

where π_j is the steady state probability distribution over the derivatives of $C_j, 1 \leq j \leq n$, and G is a normalisation constant.

The simplest situation in which a PEPA model will exhibit a product form solution with respect to its cooperating components will be when those components are completely independent of each other, i.e. when there is no cooperation between them (all cooperation sets are empty).

Unfortunately this structure rarely arises in models of real systems. Here we consider models in which the condition that there is no cooperation between the elements of the state vector is relaxed, so that a limited form of interaction is introduced. The form of the cooperating components is restricted in order to ensure that the elements correspond to quasi-reversible Markov processes. Moreover, the nature of the interaction between them ensures the quasi-reversibility of the model component considered as a whole.

4.1. Reversed and quasi-reversible processes in PEPA

We will call PEPA components whose underlying Markov process is a quasi-reversible process, when considered in isolation, *QR components*. Initially we aim to capture a class of PEPA components which are QR

*However they will not necessarily all change with every transition of the derivation graph.

and which correspond to nodes within a queueing network. In Section 5., we will demonstrate that the class of QR components is in fact more general. However these *input-output components* will provide a useful basis from which to explain the theorems in this section.

As a first step we identify activities which form the input and output processes defined in Section 2.. Recall that these are analogous to customer arrivals and departures in a queue. The crucial characteristic of these pairs of events is the fact that they *reverse* each other: one event followed immediately by the other returns us to the original state.

DEFINITION 4.1. *A PEPA component P is said to enable a* reverse pair $(\alpha, -\alpha)$, *if* $(\alpha, r) \in \mathcal{A}ct(P)$ *and for every* (α, r)*-derivative* P' ($P \xrightarrow{(\alpha,r)} P'$) *there exists* $(-\alpha, s) \in \mathcal{A}ct(P')$ *such that P is an* $(-\alpha, s)$*-derivative of* P', *where* $r, s \in \mathbb{R}^+ \cup \{\top\}$.

Note that if $(\alpha, -\alpha)$ is a reverse pair between components P and P', then $(-\alpha, \alpha)$ is a reverse pair between P' and P, i.e. $-(-\alpha) = \alpha$. For example, in the queue modelled below, $(arrive, serve)$ is a reverse pair in every derivative, but $(serve, arrive)$ is another reverse pair in all Q_i, $i \geq 1$.

$$\begin{aligned} Q_0 &\stackrel{\text{def}}{=} (arrive, \lambda).Q_1 \\ Q_i &\stackrel{\text{def}}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1} \qquad i \geq 1 \end{aligned}$$

If all the activities that a PEPA component enables constitute half of a reverse pair then it is clear how to form the reversed Markov process of the Markov process underlying the component. For any reverse pair $(\alpha, -\alpha)$, if α occurs in the original process at time t then in the reverse process $-\alpha$ occurs at time $-t$.

The simplest form of quasi-reversible PEPA processes will be those which exhibit a strict input/output behaviour with respect to their reverse pairs. As a preliminary we generalise the notion of an (α, r)-derivative.

DEFINITION 4.2. *If* $P \underbrace{\xrightarrow{(\alpha,r)} \cdots \xrightarrow{(\alpha,r)}}_{n \text{ times } (\alpha,r)} P'$ *then P' is an* $(\alpha, r)^n$-derivative *of P.*

In order to form an *input process*, corresponding to arrivals at a queue, we require that the same *input* action (the (α, r) of a reverse pair $(\alpha, -\alpha)$) is enabled in all the derivatives of the component and that the component must change state monotonically whenever such an action occurs.

DEFINITION 4.3. *A PEPA component P with initial component P_0 is an* input-output *component if P_0 enables only passive activities* $(\alpha, \top)$ *such that:*

1. $(\alpha, -\alpha)$ *is a reverse pair;*
2. $(\alpha, \top)$ *and* $(-\alpha, s)$, *for some s, are the only activities enabled by any derivative of P_0; and*
3. *the* $(\alpha, \top)^n$*-derivatives of P_0 are distinct for all n.*

The set of action types $\mathcal{I}(P) = \mathcal{A}(P_0)$ is the set of *input actions* of P. These correspond directly to the input subprocesses in the underlying Markov process: one action type for each input subprocess. The *output actions* are those which reverse the input actions:

$$\mathcal{O}(P) = \{\beta \mid \beta = -\alpha \wedge \alpha \in \mathcal{I}(P)\}$$

The completion of activities of these types corresponds to the output subprocess of the appropriate class. We assume that the input and output actions of any input-output process are disjoint.

A PEPA component containing passive activities which are not shared with an active component is incomplete. We say that a component $S_\alpha^> \stackrel{\text{def}}{=} (\alpha, r).S_\alpha^>$ such that $r \neq \top$, is an *external source* for an input-output process P if $\alpha \in \mathcal{I}(P)$.

PROPOSITION 4.1. *If a PEPA model M consists of an input-output component P in cooperation with an external source $S_\alpha^>$ for each $\alpha \in \mathcal{I}(P)$,*

$$M \stackrel{\text{def}}{=} P \underset{\mathcal{I}(P)}{\bowtie} (S_{\alpha_1}^> \parallel \cdots \parallel S_{\alpha_n}^>)$$

where $\mathcal{I}(P) = \{\alpha_1, \ldots \alpha_n\}$, then the Markov process underlying M is quasi-reversible, i.e. M is QR.

This follows from the definition of quasi-reversibility given in Section 2..

If $\mathcal{I}(P) = \{\alpha\}$ then by the definition of an input-output component, the Markov process underlying M is isomorphic to a reversible queue with one class of customers, whose arrival process is determined by the external source.

4.2. Interaction between quasi-reversible PEPA processes

Now we consider models formed by the interaction of input-output components, and their appropriate external sources. Suppose $C \stackrel{\text{def}}{=} P \underset{L}{\bowtie} Q$ where both P and Q are input-output components. We say that $\alpha \in L$ is a *channel from P to Q* if $\alpha \in \mathcal{O}(P) \cap \mathcal{I}(Q)$; similarly $\beta \in L$ is a *channel from Q to P* if $\beta \in \mathcal{I}(P) \cap \mathcal{O}(Q)$. Note that since the input and output sets of all components are distinct all channels are uni-directional.

If every action in the set $L \cap (\mathcal{I}(P) \cup \mathcal{I}(Q))$ forms a channel in the component $P \underset{L}{\bowtie} Q$, the interaction is termed a *flow cooperation*. (Any action in L but not in $\mathcal{I}(P) \cup \mathcal{I}(Q)$ is unable to participate in the cooperation, of course, but would be syntactically valid.) In the more general case, a cooperation of n input-output components $P_1, \ldots, P_n$, with state vector representation $(P_1, \ldots, P_n)$, is a *flow cooperation* if, when we consider the pairwise interactions between the components, they all form channels. More formally, for any P_i, P_j, $1 \leq i, j \leq n$ such that $i \neq j$, if P_i and P_j cooperate over α then either α is a channel from P_i to P_j or α is a channel from P_j to P_i. We denote the set of interactions between P_i and P_j by L_{ij}.

For any QR component, the input actions are those corresponding to an input subprocess in the underlying quasi-reversible process, and the output actions are those corresponding to an output subprocess. A flow cooperation between such components is then defined as above.

Note that this does not necessarily imply that all interactions are one-to-one. For example, suppose we have a component formed from input-output components:

$$P \equiv P_1 \underset{\{\alpha,\beta\}}{\bowtie} (P_2 \parallel P_3)$$

where

$$\mathcal{I}(P_1) = \{\alpha\} \qquad \mathcal{I}(P_2) = \mathcal{I}(P_3) = \{\beta\}$$
$$\mathcal{O}(P_1) = \{\beta\} \qquad \mathcal{O}(P_2) = \mathcal{O}(P_3) = \{\alpha\}$$

Here P_2 and P_3 compete for the output from P_1; similarly they both direct their output to P_1. However if we consider the pairwise interactions between components in the state vector representation (P_1, P_2, P_3), there is either no interaction between them (P_2 and P_3) or the interaction takes the form of channels in both directions.

4.2.1. "Open" interactions

As mentioned in the previous section, a model which enables passive actions is considered to be incomplete and is not well-formed (Hillston, 1994a). In the case of interacting input-output components this might arise if there are input actions which are not part of a channel. For example, consider a modification to the system above such that $\mathcal{I}(P_3)$ becomes γ while all other components remain unchanged. Now the model P enables $(\gamma, \top)$.

The obvious solution in this case, and in general, is to provide an external source of the appropriate action type. The model becomes

$$M \stackrel{\mathit{def}}{=} S_\gamma^{>} \underset{\{\gamma\}}{\bowtie} P \equiv S_\gamma^{>} \underset{\{\gamma\}}{\bowtie} (P_1 \underset{\{\alpha,\beta\}}{\bowtie} (P_2 \parallel P_3))$$

The resulting actions will form an input subprocess to the model.

Similarly, if a model enables an output action which does not form a channel the model itself will exhibit an output subprocess. To make this situation explicit we complement an unsynchronised output action with an *external sink*. This is analogous to an external source but with a passive action of the appropriate type. This additional cooperating component does not add any information to the model and is redundant in the state vector representation.

In some circumstances we may wish to model situations in which the environment competes with the cooperating components for the output of a given component. In this case we need to introduce an external sink with an appropriate weighting. For example, if, in the small system above, output from P_2 or P_3 cooperates with P_1 with probability 1/3 but is lost into the environment with probability 2/3, then we introduce an external sink $S_\alpha^{<}$, where

$$S_\alpha^{<} \stackrel{\mathit{def}}{=} (\alpha, 2\top).S_\alpha^{<}$$

In a queueing network context, a sink represents *departures* from a network.

Note that both sources and sinks may be regarded as degenerate forms of input-output components, which complete actions of only one type. In the case of a source the input action is missing; in the case of a sink, the output action. However, in either case the interaction of the source or sink with the component it complements is still a flow cooperation since a source participates in a channel from itself to the component; a sink, in a channel from the component to itself.

DEFINITION 4.4. *Consider a PEPA model M comprising a flow cooperation of input-output components $P_1, \ldots, P_n$. M is said to be* open *if it contains any external sources or sinks.*

By Proposition 4.1 a flow cooperation of input-output components, together with appropriate sources and sinks, is QR. The following generalisation of Kelly's Theorem 3.7 (Kelly, 1979) therefore holds,

THEOREM 4.2. (OPEN INTERACTIONS) *An open flow cooperation of QR components has the following properties at equilibrium:*

i) The marginal states of the individual components are independent.
ii) For an individual component the equilibrium distribution and the distribution over states at the time of completion of an input action of a given type are identical and are both as they would be if the component were in isolation with independent Poisson external sources with rates given by traffic equations.
iii) Under time reversal the model becomes another flow cooperation of QR components; in particular sources become sinks, and vice versa.
iv) The underlying Markov process is quasi-reversible. Thus, the completion of output actions of each type at the sinks form independent Poisson processes and the state of the model at any time t_0 is independent of the completion instants of output actions in the model prior to time t_0.

4.2.2. "Closed" interactions

It is not difficult to see that in some cases external sources or sinks will not be necessary to complete a model containing the flow cooperation of input-output components. If all the input action needs of the cooperating components are supplied by output actions of other components in the cooperation, and similarly all output actions produced by components are shared, as input actions, with other components, it is clear that the model is self-sufficient in this sense.

DEFINITION 4.5. *A flow cooperation is said to be* closed *if it has no actions enabled which do not form a channel under the cooperation.*

For closed PEPA models which are flow cooperations of QR components we have the following generalisation of Kelly's Theorem 3.12 (Kelly, 1979), analogous to that given for an open PEPA model in the previous section.

THEOREM 4.3. (CLOSED INTERACTIONS) *A closed flow cooperation of QR components $C_1, \ldots, C_m$ has the following properties at equilibrium:*

i) Let $\pi(C_{k_i})$ be the equilibrium probability that component C_k is in state C_{k_i}, $1 \leq k \leq m$, when interacting only with its Poisson external sources with rates given by the traffic equations—i.e. as in ii) of Theorem 4.2. Then the equilibrium state probability distribution of the closed flow cooperation is

$$\prod_{i=1}^{m} \pi(C_{k_i})/G$$

where G is the normalising constant.

ii) The distribution over states at the time of completion of an input action of type α, which produces an $(\alpha, r)^n$-derivative in a component of the state vector representation, is identical to that of i) for the closed flow cooperation obtained by reducing n to $n-1$.

iii) Under time reversal the model becomes another closed flow cooperation of QR components.

Note that the resulting process is no longer QR. All closed QR processes are built by successively combining QR processes via *open* flow cooperations, until the last step which closes the flow cooperation. The closed process can no longer participate as a component in a larger flow cooperation.

4.3. A simple example

In this section we consider in more detail the example suggested at the end of Section 2. We will consider a packet-switched message transmission system. Messages of varying length arrive from a sender and must be sent through the network to the receiver. Each message is broken up into a number of packets depending on its length, and these are transmitted between nodes as a packet train.

If we consider movement of packet trains with differing numbers of packets as different action types at each link we can model the network as a flow cooperation of input-output components with one such component for each node. The sender corresponds to a collection of external sources, and the receiver, to a collection of sinks, one of each for the different possible message lengths (Figure 1).

For simplicity here we consider only two types of messages, those which result in *long* packet trains and those which result in *short* ones. However, it is easy to see how more general classes of messages could be obtained in the same way.

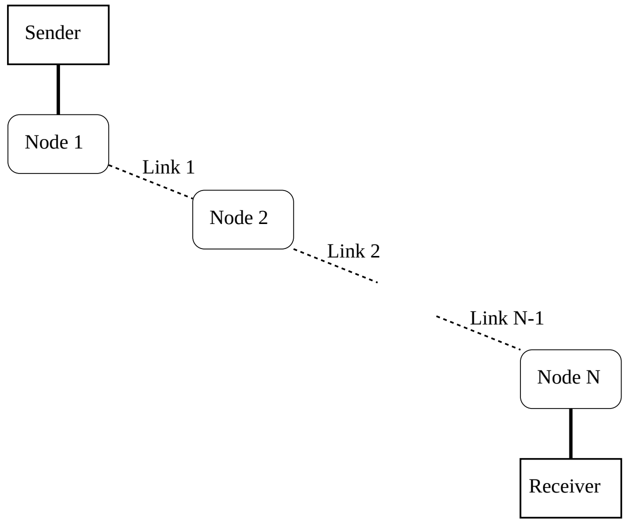

FIGURE 1. A simple packet transmission network

For each type of message, each link in the network will be represented by two actions in the model: an input action in the receiving node and an output action in the transmitting node. Conversely, each node will be represented as an input-output component with one action type for each type of message. We assume that node k can transmit short messages at rate μ_{k1}, and long messages at rate μ_{k2}. Thus the first node of the network, $\mathsf{N}1$, which receives messages from the sender and then transmits them down the first link is modelled as follows, where snd_{sht} and snd_{lg} are the actions of sending short and long packet trains respectively. The actions lk^1_{sht} and lk^1_{lg} represent the transmissions of short and long packet trains down the first link.

$$\begin{aligned}
\mathsf{N}1_{0,0} &\stackrel{def}{=} (snd_{sht}, \top).\mathsf{N}1_{1,0} + (snd_{lg}, \top).\mathsf{N}1_{0,1} \\
\mathsf{N}1_{i,0} &\stackrel{def}{=} (snd_{sht}, \top).\mathsf{N}1_{i+1,0} + (snd_{lg}, \top).\mathsf{N}1_{i,1} \\
&\quad + (lk^1_{sht}, \mu_{11}).\mathsf{N}1_{i-1,0} \\
\mathsf{N}1_{0,j} &\stackrel{def}{=} (snd_{sht}, \top).\mathsf{N}1_{1,j} + (snd_{lg}, \top).\mathsf{N}1_{0,j+1} \\
&\quad + (lk^1_{lg}, \mu_{12}).\mathsf{N}1_{0,j-1} \\
\mathsf{N}1_{i,j} &\stackrel{def}{=} (snd_{sht}, \top).\mathsf{N}1_{i+1,j} + (snd_{lg}, \top).\mathsf{N}1_{i,j+1} \\
&\quad + (lk^1_{sht}, \mu_{11}).\mathsf{N}1_{i-1,j} + (lk^1_{lg}, \mu_{12}).\mathsf{N}1_{i,j-1}
\end{aligned}$$

Similarly the final node of the network receives messages down the final link and then delivers them to the receiver (actions $dlvr_{sht}$ and $dlvr_{lg}$ respectively):

$$\begin{aligned}
\mathsf{N}N_{0,0} &\stackrel{def}{=} (lk^{N-1}_{sht}, \top).\mathsf{N}N_{1,0} + (lk^{N-1}_{lg}, \top).\mathsf{N}N_{0,1} \\
\mathsf{N}N_{i,0} &\stackrel{def}{=} (lk^{N-1}_{sht}, \top).\mathsf{N}N_{i+1,0} + (lk^{N-1}_{lg}, \top).\mathsf{N}N_{i,1} \\
&\quad + (dlvr_{sht}, \mu_{N1}).\mathsf{N}N_{i-1,0}
\end{aligned}$$

$$\begin{aligned}
\mathsf{N}N_{0,j} &\stackrel{def}{=} (lk_{sht}^{N-1}, \top).\mathsf{N}N_{1,j} + (lk_{lg}^{N-1}, \top).\mathsf{N}N_{0,j+1} \\
&\quad + (dlvr_{lg}, \mu_{N2}).\mathsf{N}N_{0,j-1} \\
\mathsf{N}N_{i,j} &\stackrel{def}{=} (lk_{sht}^{N-1}, \top).\mathsf{N}N_{i+1,j} + (lk_{lg}^{N-1}, \top).\mathsf{N}N_{i,j+1} \\
&\quad + (dlvr_{sht}, \mu_{N1}).\mathsf{N}N_{i-1,j} \\
&\qquad + (dlvr_{lg}, \mu_{N2}).\mathsf{N}N_{i,j-1}
\end{aligned}$$

An arbitrary node in the middle of the network will receive a packet train from its incoming link, lk^{k-1}, and then pass it on down its outgoing link, lk^k.

$$\begin{aligned}
\mathsf{N}k_{0,0} &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}k_{1,0} + (lk_{lg}^{k-1}, \top).\mathsf{N}k_{0,1} \\
\mathsf{N}k_{i,0} &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}k_{i+1,0} + (lk_{lg}^{k-1}, \top).\mathsf{N}k_{i,1} \\
&\quad + (lk_{sht}^{k}, \mu_{k1}).\mathsf{N}k_{i-1,0} \\
\mathsf{N}k_{0,j} &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}k_{1,j} + (lk_{lg}^{k-1}, \top).\mathsf{N}k_{0,j+1} \\
&\quad + (lk_{lg}^{k}, \mu_{k2}).\mathsf{N}k_{0,j-1} \\
\mathsf{N}k_{i,j} &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}k_{i+1,j} + (lk_{lg}^{k-1}, \top).\mathsf{N}k_{i,j+1} \\
&\quad + (lk_{sht}^{k}, \mu_{k1}).\mathsf{N}k_{i-1,j} + (lk_{lg}^{k}, \mu_{k2}).\mathsf{N}k_{i,j-1}
\end{aligned}$$

For all the nodes ($1 \leq k \leq N$), the derivative $\mathsf{N}k_{i,j}$ captures the state of the node when it contains i short messages waiting for transmission, and j long ones. We assume that the buffering capacity of all nodes is unbounded from a practical point of view.

The network is then the flow cooperation of all these components:

$$Network \stackrel{def}{=} \mathsf{N}1_{0,0} \underset{lk^1}{\bowtie} \cdots \underset{lk^{k-1}}{\bowtie} \mathsf{N}k_{0,0} \underset{lk^k}{\bowtie} \cdots \underset{lk^{N-1}}{\bowtie} \mathsf{N}N_{0,0}$$

where lk^i denotes $\{lk_{sht}^i, lk_{lg}^i\}$. The sender is represented as two source components, one for short messages and one for long messages, which are independent of each other:

$$\begin{aligned}
S_{snd_{sht}}^{>} &\stackrel{def}{=} (snd_{sht}, \lambda_s).S_{snd_{sht}}^{>} \\
S_{snd_{lg}}^{>} &\stackrel{def}{=} (snd_{lg}, \lambda_l).S_{snd_{lg}}^{>} \\
Sender &\stackrel{def}{=} S_{snd_{sht}}^{>} \parallel S_{snd_{lg}}^{>}
\end{aligned}$$

Similarly we represent the receiver as two independent sinks, one for each type of message:

$$\begin{aligned}
S_{dlvr_{sht}}^{<} &\stackrel{def}{=} (dlvr_{sht}, \top).S_{dlvr_{sht}}^{<} \\
S_{dlvr_{lg}}^{<} &\stackrel{def}{=} (dlvr_{lg}, \top).S_{dlvr_{lg}}^{<} \\
Receiver &\stackrel{def}{=} S_{dlvr_{sht}}^{<} \parallel S_{dlvr_{lg}}^{<}
\end{aligned}$$

The complete system is modelled as the cooperation of the *Sender*, *Receiver* and *Network* components:

$$System \stackrel{def}{=} Sender \underset{\{snd_{sht}, snd_{lg}\}}{\bowtie} Network \underset{\{dlvr_{sht}, dlvr_{lg}\}}{\bowtie} Receiver$$

This is clearly a flow cooperation of QR components and we can apply Theorem 4.2 to deduce that in equilibrium each node behaves as if independent (with its own external source). Thus the steady state probability that the system is in some state

$$(\mathsf{N}1_{i_1,j_1}, \ldots, \mathsf{N}k_{i_k,j_k}, \ldots, \mathsf{N}N_{i_N,j_N}) \tag{1}$$

is given by the product

$$\prod_{k=1}^{N} \Pi_k \left(S_{lk^{k-1}}^{>} \underset{lk^{k-1}}{\bowtie} \mathsf{N}k_{i_k,j_k} \right)$$

where lk^{k-1} denotes $\{lk_{sht}^{k-1}, lk_{lg}^{k-1}\}$ and Π_k is the steady state distribution over the kth node in isolation with Poisson arrivals provided by an external source with appropriate rates:

$$\begin{aligned}
S_{lk^{k-1}}^{>} &\stackrel{def}{=} S_{lk_{sht}^{k-1}}^{>} \parallel S_{lk_{lg}^{k-1}}^{>} \\
S_{lk_{sht}^{k-1}}^{>} &\stackrel{def}{=} (lk_{sht}^{k-1}, \nu_{s,k}).S_{lk_{sht}^{k-1}}^{>} \\
S_{lk_{lg}^{k-1}}^{>} &\stackrel{def}{=} (lk_{lg}^{k-1}, \nu_{l,k}).S_{lk_{lg}^{k-1}}^{>}
\end{aligned}$$

Here $\nu_{s,k}$ and $\nu_{l,k}$ are the completion rate of short and long message transfers arriving at the kth node in equilibrium. Using standard traffic arguments we can deduce that these are exactly λ_s and λ_l respectively, for all $k, 1 \leq k \leq N$, if the system is stable, i.e. $\frac{\lambda_s}{\mu_{k1}} + \frac{\lambda_l}{\mu_{k2}} < 1$ for all k.

If we consider the definition of the behaviour of the kth node, for arbitrary k, we can see that it is made up of two independent aspects: the behaviour with respect to short messages and the behaviour with respect to long messages. This suggests an alternative representation of $\mathsf{N}k$ as follows:

$$\begin{aligned}
\mathsf{N}k'_{0,0} &\stackrel{def}{=} \mathsf{N}_S k_0 \parallel \mathsf{N}_L k_0 \\
\mathsf{N}_S k_0 &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}_S k_1 \\
\mathsf{N}_S k_i &\stackrel{def}{=} (lk_{sht}^{k-1}, \top).\mathsf{N}_S k_{i+1} \; + \; (lk_{sht}^{k}, \mu_{k1}).\mathsf{N}_S k_{i-1} \\
\mathsf{N}_L k_0 &\stackrel{def}{=} (lk_{lg}^{k-1}, \top).\mathsf{N}_L k_1 \\
\mathsf{N}_L k_j &\stackrel{def}{=} (lk_{lg}^{k-1}, \top).\mathsf{N}_L k_{j+1} \; + \; (lk_{lg}^{k}, \mu_{k2}).\mathsf{N}_L k_{j-1}
\end{aligned}$$

It is straightforward to verify that $\mathsf{N}k'_{0,0}$ is isomorphic to $\mathsf{N}k_{0,0}$. Moreover, using the equational laws introduced in (Hillston, 1994a), we can manipulate the terms as shown below, using the equational laws for cooperation:

$$\begin{aligned}
& S_{lk^{k-1}}^{>} \underset{lk^{k-1}}{\bowtie} \mathsf{N}k_{i_k,j_k} \equiv \\
& \left(S_{lk_{sht}^{k-1}}^{>} \parallel S_{lk_{lg}^{k-1}}^{>} \right) \underset{lk^{k-1}}{\bowtie} (\mathsf{N}_S k_{i_k} \parallel \mathsf{N}_L k_{j_k}) = \\
& \left(S_{lk_{sht}^{k-1}}^{>} \underset{lk_{sht}^{k-1}}{\bowtie} \mathsf{N}_S k_{i_k} \right) \underset{lk^{k-1}}{\bowtie} \left(S_{lk_{lg}^{k-1}}^{>} \underset{lk_{lg}^{k-1}}{\bowtie} \mathsf{N}_L k_{j_k} \right) = \\
& \qquad \left(S_{lk_{sht}^{k-1}}^{>} \underset{lk_{sht}^{k-1}}{\bowtie} \mathsf{N}_S k_{i_k} \right) \parallel \left(S_{lk_{lg}^{k-1}}^{>} \underset{lk_{lg}^{k-1}}{\bowtie} \mathsf{N}_L k_{j_k} \right)
\end{aligned}$$

where the last equality holds because the activity sets of the two pairs of components in parentheses are disjoint, i.e. only short messages are treated on the left and only long messages are treated on the right. In this form it is apparent that the behaviours with respect to the long and short messages are independent of each other. This implies that the equilibrium distribution of the kth node has a product form with respect to these independent aspects of its behaviour. Thus,

$$\Pi_k\left(S^{>}_{lk^{k-1}} \underset{lk^{k-1}}{\bowtie} \mathsf{N}k_{i_k,j_k}\right) = \left(\frac{\lambda_s}{\mu_{k1}}\right)^{i_k}\left(\frac{\lambda_l}{\mu_{k2}}\right)^{j_k}\left(1-\frac{\lambda_s}{\mu_{k1}}\right)\left(1-\frac{\lambda_l}{\mu_{k2}}\right)$$

It follows that the steady state probability that the system is in the state shown in expression 1 is

$$\prod_{k=1}^{N}\left[\left(\frac{\lambda_s}{\mu_{k1}}\right)^{i_k}\left(\frac{\lambda_l}{\mu_{k2}}\right)^{j_k}(1-\frac{\lambda_s}{\mu_{k1}})(1-\frac{\lambda_l}{\mu_{k2}})\right]$$

5. MORE COMPLEX EXAMPLES

In this section we consider two examples which are more complex than the simple packet transmission system presented in Section 4.3. The first, although still based on input-output components, contains a component which does not correspond to a server in a conventional queueing network. This fault tolerant server, since it is isomorphic to an input-output component, is QR by Proposition 4.1. The second example is based on components even further removed from the notion of a service centre: these components model rides within a fairground. They are not input-output components but we show that they are QR, i.e. the Markov process underlying each component is quasi-reversible and the input actions are easily identified. Assuming a fixed population of revellers who circulate between the different rides within the fairground we derive a product form expression for the equilibrium state of the fairground in terms of the number of riders on each ride.

5.1. A fault tolerant processor

Consider a server which has many states (Figure 2); we assume it is a fault tolerant processor which can maintain a constant level of service under many minor faults before catastrophic failure occurs—so many levels that we may regard them as infinite. When the processor has suffered a failure it immediately starts a repair procedure to return it to its previous state. However, in the meantime further faults might occur. Jobs arrive at the processor in a Poisson arrival stream with constant rate determined by the environment. We assume that no jobs are lost when failures or repairs occur.

This simple system is defined in PEPA as below. The pairs of activities $(in, \top)$ and (out, μ), and $(fail, \xi)$ and (fix, ν) form reverse pairs. Since the activity $(fail, \xi)$ is not passive P_{10} does not satisfy the conditions of Definition 4.3, i.e. it is not an input-output component. However it is staightforward to see that it is isomorphic to the flow cooperation of a similar component in which the action $fail$ is passive, and a source component for $fail$ with rate ξ. It follows that P_{10} is a QR component.

$$\begin{array}{rcll}
P_{10} & \stackrel{def}{=} & (in,\top).P_{11} + (fail,\xi).P_{20} & \\
P_{1i} & \stackrel{def}{=} & (in,\top).P_{1i+1} + (out,\mu).P_{1i-1} & \\
 & + & (fail,\xi).P_{2i} & i \geq 1 \\
P_{j0} & \stackrel{def}{=} & (in,\top).P_{j1} + (fail,\xi).P_{j+10} & \\
 & + & (fix,\nu).P_{j-10} & j \geq 2 \\
P_{ji} & \stackrel{def}{=} & (in,\top).P_{ji+1} + (out,\mu).P_{ji-1} & \\
 & + & (fail,\nu).P_{j+1i} + (fix,\nu).P_{j-1i} & i \geq 1, j \geq 2
\end{array}$$

It is straightforward to verify that, when the processor is considered in a flow cooperation with an appropriate source,

$$S^{>}_{in} \stackrel{def}{=} (in, \lambda_{in}).S^{>}_{in}$$

the steady state probability that the processor is in state P_{ji} is

$$\Pi(P_{ji}) = \left(\frac{\lambda_{in}}{\mu}\right)^i\left(\frac{\xi}{\nu}\right)^j\left(1-\frac{\lambda_{in}}{\mu}\right)\left(1-\frac{\xi}{\nu}\right)$$

We suppose that jobs are preprocessed before submission to the processor, and that the preproccessor behaves as a simple $M/M/1$ queue. Jobs arrive at the preprocessor in a Poisson stream, are processed in FCFS order and then served and passed to the processor (with rate μ). The PEPA specification of a preprocessor is as follows:

$$\begin{array}{rcll}
PP_0 & \stackrel{def}{=} & (arrive,\top).PP_1 & \\
PP_i & \stackrel{def}{=} & (arrive,\top).PP_{i+1} + (in,\lambda_{in}).PP_{i-1} & i \geq 1
\end{array}$$

The steady state probability, when the preprocessor is in flow cooperation with the source

$$S^{>}_{arrive} \stackrel{def}{=} (arrive, \lambda_{arr}).S^{>}_{arrive}$$

with $\lambda_{arr} < \lambda_{in}$, is

$$\Pi(PP_k) = \left(\frac{\lambda_{arr}}{\lambda_{in}}\right)^k\left(1-\frac{\lambda_{arr}}{\lambda_{in}}\right)$$

The system is then the open interaction of the preprocessor and the processor:

$$System \stackrel{def}{=} PP_0 \underset{\{in\}}{\bowtie} P_{10}$$

which is clearly a flow cooperation. Assuming that the arrival stream is provided by the source $S^{>}_{arrive}$ defined above we can deduce that the steady state probability that the system is in state (PP_k, P_{ji}) is

$$\Pi(PP_k, P_{ji}) = \Pi(PP_k)\,\Pi(S^{>}_{in} \underset{\{in\}}{\bowtie} P_{ji})$$

$$= \left(\frac{\lambda_{arr}}{\lambda_{in}}\right)^k\left(\frac{\lambda_{in}}{\mu}\right)^i\left(\frac{\xi}{\nu}\right)^j\left(1-\frac{\lambda_{arr}}{\lambda_{in}}\right)\left(1-\frac{\lambda_{in}}{\mu}\right)\left(1-\frac{\xi}{\nu}\right)$$

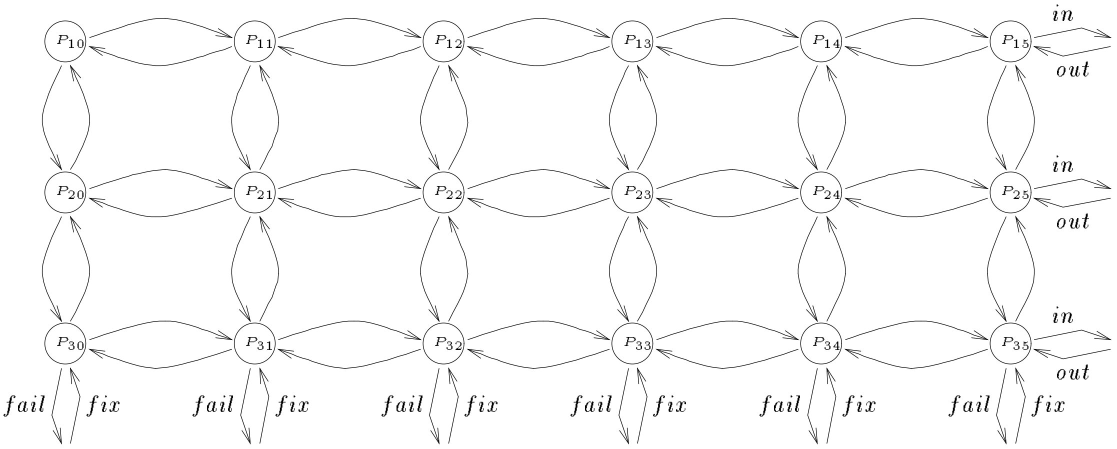

FIGURE 2. State transition diagram of the fault tolerant processor, P_{10}

5.2. Fairground rides

We consider a fairground, F, which is made up of a series of rides, R_k, $1 \leq k \leq N$. We assume that there is a fixed population of riders, M, within the fairground who circulate between the rides. We will show that, under appropriate assumptions, each ride R_k corresponds to a quasi-reversible process when we consider it in isolation, with the arrivals and departures of riders forming the input and output processes respectively. Thus, since the fairground is defined as a closed flow cooperation of rides, we can apply Theorem 4.3 to deduce that

$$\Pi(F) = \frac{1}{G} \prod_{k=1}^{N} \pi(R_k)$$

However, the behaviour of each R_k cannot be modelled as a service centre in a queueing network.

After a rider joins a ride numbered k, he is erratically moved clockwise ($spin$) and anti-clockwise ($back$) between different stages S_{ki} of the ride at rates c_{ki} and a_{ki} respectively, where i ranges from 1 to the number of stages in ride k. The rider may join the ride, or leave it, at any stage; departures from each stage occur at rate μ_k. The capacity of the ride is unbounded up to the population of the fairground and the current riders move individually between the different stages. When considering a single ride in isolation, we assume that the arrivals of riders form a Poisson stream with rate λ_k. For simplicity, here we assume there are just three stages in the ride; we also assume that

$$c_{k1}c_{k2}c_{k3} = a_{k1}a_{k2}a_{k3} \qquad (2)$$

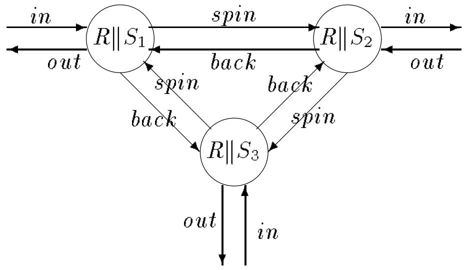

FIGURE 3. Fragment of the state transition diagram of the fairground ride with one rider

in order to establish the QR property for each ride k (see below). The PEPA model of the kth ride, R_k is given below:

$$\begin{aligned}
R_k &\stackrel{\text{def}}{=} (in_k, \top).(S_1 \parallel R_k) + (in_k, \top).(S_2 \parallel R_k) \\
&\qquad + (in_k, \top).(S_3 \parallel R_k) \\
S_1 &\stackrel{\text{def}}{=} (spin, c_{k1}).S_2 + (back, a_{k1}).S_3 + (out, \mu).\mathbf{0} \\
S_2 &\stackrel{\text{def}}{=} (spin, c_{k2}).S_3 + (back, a_{k2}).S_1 + (out, \mu).\mathbf{0} \\
S_3 &\stackrel{\text{def}}{=} (spin, c_{k3}).S_1 + (back, a_{k3}).S_2 + (out, \mu).\mathbf{0}
\end{aligned}$$

where $\mathbf{0}$ is the component which can carry out no actions. The input process is represented by three separate actions because it is non-deterministic at which of the three stages a new rider will start. The fragment of the derivation graph corresponding to the derivatives when there is just one rider is shown in Figure 3.

The Markov process that models a ride considered in isolation is reversible by the Kolmogorov criteria, since the transition rates satisfy equation 2 and the instantaneous departure rates are the same from all stages. We take the arrivals of riders (in) as the input process and the departure of riders (*out*) as the output process, and consider the state of the ride to be the numbers of riders currently present in each stage—a triple (n_{k1}, n_{k2}, n_{k3}). It is clear from the PEPA definition that at an arbitrary time t_0 the state of the ride is independent of the input process after t_0. But since the Markov model is reversible, we can deduce that the state of the ride is also independent of the output process prior to time t_0. Thus R_k is a QR component, as required.

The state of the whole fairground is described by $\boldsymbol{f} = ((n_{11}, n_{12}, n_{13}), \ldots, (n_{N1}, n_{N2}, n_{N3}))$, a vector of triples, where the sum of all of the components of all of the triples is M. For ride k in isolation, the equilibrium state probability distribution is given by:

$$\pi_k(n_{k1}, n_{k2}, n_{k3}) = \left(\frac{\lambda_k}{\mu_k}\right)^{n_{k1}+n_{k2}+n_{k3}} \left(\frac{c_{k3}}{a_{k1}}\right)^{n_{k1}} \left(\frac{a_{k3}}{c_{k2}}\right)^{n_{k2}}$$

The equilibrium probability distribution for the state of the whole fairground is therefore

$$\pi(\boldsymbol{f}) = \frac{1}{G} \prod_{k=1}^{N} \left(\frac{e_k}{\mu_k}\right)^{n_{k1}+n_{k2}+n_{k3}} \left(\frac{c_{k3}}{a_{k1}}\right)^{n_{k1}} \left(\frac{a_{k3}}{c_{k2}}\right)^{n_{k2}}$$

where e_k is the visitation rate of riders to ride k, arbitrary up to a multiplicative constant, i.e. the proportion of all his rides that a rider takes at ride k.

6. CONCLUSIONS

We have established a link between the compositional structure of MPA models and the independent analysis of components based on quasi-reversibility. In particular we have shown that when the interactions between suitable components are limited to be *flow cooperations* the components behave as if independent (up to normalisation). Thus the steady state behaviour of such an interaction can be expressed as a product of the behaviours of the components considered in isolation. Additionally, in the case of *open* models, the model itself is a "QR" component, giving rise to a quasi-reversible Markov process.

We have identified a class of such QR components which can be readily recognised syntactically, the *input-output* components, and demonstrated that with this class of components and flow cooperation we can capture systems which cannot be modelled by conventional queueing networks. However, as demonstrated by the final example in the paper, the class of QR components is more substantial than the class of input-output components. We hope to develop a syntactic characterisation of QR components in the future.

There are several interesting directions for further work. Although a basis for product form solutions has been established, taking advantage of the structure of MPA models, several practical issues must be resolved before an efficient implementation of such a solution technique can be developed. For example, we have yet to show how the independent behaviour of components can be derived in the general case. This will involve characterising the influence of the rest of the model on each component. In particular, deriving the activity rates for the necessary external sources has not been addressed in this paper. These must capture the completion rate of input activities, analogous to the arrival rate of customers in queueing networks, when the system is in equilibrium. This suggests that the necessary information could be derived from *event equations*, analogous to the traffic equations of queueing networks. This problem has been considered in (Sereno, 1995). In the case of closed models, the calculation of the normalising constant will also need to be considered.

The class of models which exhibit quasi-reversible structures will always be small. Ultimately the strength of this methodology lies in its potential use as a bounding technique. Future work will also involve the development of precongruence relations for Markovian process algebras. These relations will allow a model to be formally manipulated into a form which exhibits the quasi-reversible structure and is therefore amenable to efficient product form solution.

Acknowledgements

Jane Hillston is supported by EPSRC Post-doctoral Fellowship B/93/RF/1729. Peter Harrison is partially supported by the CEC BRA project no.7269 "QMIPS".

References

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

Harrison, P.G., & Patel, N.M. 1992. *Performance Modelling of Communication Networks and Computer Architectures.* Addison-Wesley.

Hillston, J. 1994a (Apr.). *A Compositional Approach to Performance Modelling.* Ph.D. thesis, Department of Computer Science, University of Edinburgh.

Hillston, J. 1994b. The Nature of Synchronisation. *In:* Herzog, U., & Rettelbach, M. (eds), *Proc. of 2nd Process Algebra and Performance Modelling Workshop.*

Hillston, J. 1995. Compositional Markovian Modelling Using a Process Algebra. *In:* Stewart, W.J. (ed), *Numerical Solution of Markov Chains.* Kluwer.

Kelly, F. 1979. *Reversibility and Stochastic Processes.* Wiley.

Sereno, M. 1995. Towards a Product Form Solution of Stochastic Process Algebras. *The Computer Journal*, **this issue**.