A Syntactical Analysis of Reversible PEPA Models

Jane Hillston* Nigel Thomas

$\{jeh, nat\}$@dcs.ed.ac.uk

Abstract

Product form solutions have played an important role in the development of performance modelling and in particular in the popularity of queueing network models. The notions of reversibility and quasi-reversibility underpin many product form results in queueing theory. Stochastic process algebras (SPAs) support a compositional approach to performance modelling without severe restriction on the form of the underlying Markov chain. The textual nature of SPAs makes them ideal for large scale tool support for both the structural analysis and numerical solution of models. Clearly, therefore, there is substantial potential and great advantage in developing product form solutions for classes of SPA models.
In this paper we identify a class of PEPA components that exhibit reversibility and show how these components can be combined. Although developed here in the context of PEPA the results presented can be easily generalised to any of the other stochastic process algebra languages.

keywords: Markovian process algebra; reversibility; product form solutions

1 Introduction

Initial work on stochastic process algebras (SPAs) concentrated primarily on expressiveness, application and tool support; the benefits of the compositional structure within SPAs was investigated mainly in the context of model construction and simplification. However, SPAs have now reached a level of maturity whereby it is no longer sufficient to consider only how models are to be constructed, but also how they can be solved in an efficient and timely manner. More recently several preliminary results have shown that in many cases there is a clear relationship between decompositional techniques and SPA model descriptions. For example, results have been derived based on tensor algebra by Buchholz [3] and Rettelbach and Siegle [17], the application of time scale decomposition techniques by Hillston and Mertsiotakis [11] and Mertsiotakis [15], throughput approximation by Mertsiotakis and Silva [16] and, of course, product form.

Several papers have appeared which attempt to derive product form solutions for SPA models. An approach used by Lazar and Robertazzti [14], and generalised by Boucherie [2], to consider a class of stochastic Petri nets (SPNs) with competition over resources has been considered in the context of SPA in two papers by Hillston [9] and Hillston and Thomas [12]. In addition, an alternative approach to derive product form solutions in SPA models by applying earlier results for SPN has been taken by Sereno [18]. The class of components which may be used within product form models is found to be restricted, although these restrictions are now expressed in terms of the actions of the model and how they are distributed within the components. The notions of reversibility and quasi-reversibility underpin many product form results in queueing network models, see Kelly [13] for example. The case of quasi-reversibility in SPA models was considered by Harrison and Hillston [5] and the more restricted case of reversibility was considered by Bhabuta et al [1].

The characterisation presented in [1] is at the state space level and is already well known (see [13, 19]). The real value of detecting reversible structures in SPAs is to address the problem of state space explosion and to facilitate more efficient numerical solution. This can only realistically be done if these structures are detected at the syntactic level, rather than the state space level. It is this problem that we address in Section 4 by defining a class of reversible PEPA components by analogy to input-output processes in

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

queueing network models. An interesting question that is raised in [1] is that if you have reversible PEPA components how and when can you combine them whilst retaining the reversibility property? This point is tackled in Section 5 and illustrated through some examples in Section 6. Some final comments and conclusions are presented in Section 7. We begin with a brief description of reversibility followed by an outline of some properties of the Markovian SPA, PEPA.

2 Reversibility

Reversibility is a strong property which, when satisfied, greatly simplifies the solution of Markov processes[1]. In simple terms a reversible process is one which behaves identically regardless of the direction of flow (of time). More formally, a reversible process can be defined in terms of its states (see Kelly [13]).

Definition 2.1 (Reversible Process) *A stochastic process* $X(t)$ *is* reversible *if* $(X(t_1), X(t_2), \cdots, X(t_n))$ *has the same distribution as* $(X(\tau - t_1), X(\tau - t_2), \cdots, X(\tau - t_n))$ *for all* $t_1, t_2, \cdots, t_n, \tau \in \mathbb{R}$.

It follows from Definition 2.1 that the reversible process $X(t)$ is irreducible, stationary and satisfies the detailed balance equations

$$\pi(j)q(j,k) = \pi(k)q(k,j) \tag{2.1}$$

where $q(j,k)$ is the instantaneous transition rate from state j to state k and $\pi(j)$ is the steady state probability that the process is in state j, such that $\sum_{\forall j} \pi(j) = 1$. The vector π denotes the *equilibrium distribution* of the process $X(t)$.

An alternative definition of a reversible process is given by Kolmogorov's criteria:

$$q(i_1,i_2)q(i_2,i_3)\cdots q(i_{n-1},i_n)q(i_n,i_1) = q(i_1,i_n)q(i_n,i_{n-1})\cdots q(i_3,i_2)q(i_2,i_1) \tag{2.2}$$

for all finite sequences of states $i_1, i_2, i_3, \cdots, i_{n-1}, i_n$. This has the advantage of not requiring knowledge of the equilibrium distribution. Clearly, for a process to be reversible, for every non-zero instantaneous transition rate from state i_1 to state i_2, $q(i_1,i_2)$, there must be a corresponding non-zero instantaneous transition rate in the opposite direction, i.e. $q(i_2,i_1) > 0$.

An important property of reversible processes is that given two reversible processes, $X(t)$ and $Y(t)$, their product, $X(t), Y(t)$ is also reversible.

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 $i \longrightarrow i'$, or via a dual *output process* that causes a similarly specified set of changes of state in the opposite direction, for example $i' \longrightarrow i$. We will refer to such a stochastic process, $X(t)$, as an *input-output process.* In queueing network theory, the input process corresponds to arrivals to a queue and the output process to departures; the different types correspond to different customer classes and/or different routes within the network. In the simplest case there is only one type of transition in the input and output processes respectively, and such a process $X(t)$ is a *birth-death process.* That is, the state of the system is represented by an integer, i, which may increase or decrease by one at a time. Therefore the state of the system is the size of a population, an increase of one is a 'birth' and a decrease of one is a 'death'. Thus the input-output process we have described here may be thought of as a general birth-death process where the change of state is not restricted to a unitary change of a single counter.

By definition, a stationary input-output Markov process satisfies the condition that for every non-zero instantaneous transition rate from state i_1 to state i_2, $q(i_1,i_2)$, there must be a corresponding non-zero instantaneous transition rate in the opposite direction, i.e. $q(i_2,i_1) > 0$. Furthermore we can say that all reversible Markov processes are stationary input-output Markov processes. However, the condition that a process is a stationary input-output Markov process is not sufficient for it to also be a reversible Markov process.

In order to explore the set of input-output processes which are reversible we consider a set of restricted cases:

1. **Birth-death processes** : $X(t)$ is a stationary input-output Markov process, whose state space, $\mathcal{S}$, is such that every state, $i \in \mathcal{S}$, has transitions to at most two other states. It follows that since $X(t)$ is irreducible it is also strictly linear, i.e. there is no branching. If $\mathcal{S}$ is finite then it follows that there are exactly two states i_0 and i_N which have non-zero instantaneous transition

[1] The arguments in this section apply to continuous time Markov processes with a discrete state space. For an explanation of reversibility for other forms of stochastic process the interested reader is directed to Whittle [19].

rates to only one other state; these are the *bounding states* of $X(t)$. If $\mathcal{S}$ is infinite then one, or both, of the bounding states, i_0 and i_N will not be defined. All states $i \in \mathcal{S}$ which are not bounding states have non-zero instantaneous transition rates to exactly two other states. $X(t)$ clearly satisfies Kolmogorov's criteria for reversibility (2.2).

2. **Linear input-output processes** : Again we assume that $X(t)$ is a stationary input-output Markov process with state space $\mathcal{S}$. Every state, $i \in \mathcal{S}$, has transitions to a finite number of other states which form the set $\mathcal{L}_i$. For every state $j \in \mathcal{L}_i$ the transition $i \longrightarrow j$ must eventually be followed by the transition $j \longrightarrow i$ for the process $X(t)$ to return to state i. An alternative definition is that if a transition from state i to state j is possible, i.e. $q(i,j) > 0$, then state i and state j communicate only through that transition.

3. **Parallel linear input-output processes** : A further class of reversible input-output processes can be identified by applying the property that if both $X(t)$ and $Y(t)$ are reversible input-output Markov processes then their product process $Z(t) = (X(t), Y(t))$ is a reversible input-output Markov process. Clearly $Z(t)$ is not a linear input-output process.

4. **Symmetrical branch-join processes** : Again we assume that $X(t)$ is a stationary input-output Markov process with state space $\mathcal{S}$. Every state, $i \in \mathcal{S}$, has transitions to a finite number of other states, denoted by the set $\mathcal{M}_i$. A subset of these states, $\mathcal{L}_i \subset \mathcal{M}_i$, behave exactly as in a linear input-output process. That is, for every state $j \in \mathcal{L}_i$ the transition $i \longrightarrow j$ must eventually be followed by the transition $j \longrightarrow i$ for the process $X(t)$ to return to state i. The remaining states in $\mathcal{M}_i$ are denoted by $\mathcal{K}_i$, where $\mathcal{K}_i \cap \mathcal{L}_i = \emptyset$ and $|\mathcal{K}_i| \geq 2$. The subset $\mathcal{K}_i$ can be further divided into n subsets $\mathcal{K}(k)_i$, $k = 1 \cdots n$, each having at least two members, where every state $j \in \mathcal{K}(k)_i$ is symmetrical to every other state in $\mathcal{K}(k)_i$. For two states, i and j, to be symmetrical we require that every pair of non-zero instantaneous transition rates, $q(i,i')$ and $q(i',i)$, is matched by an identical pair of rates, $q(j,j')$ and $q(j',j)$ such that either $i' = j'$, or i' and j' are symmetrical. (Note that it is possible for $i' = j$ and $j' = i$.)

Obviously birth-death processes are a subset of linear input-output processes which are a subset of symmetrical branch-join processes. A further class can be identified by considering products of symmetrical branch-join processes. There are reversible input-output Markov processes which do not belong to any of the cases identified here, for example certain classes of asymmetrical branch-join processes. However, for the remainder of this paper we will restrict ourselves to the class of processes defined here.

3 PEPA

This section briefly overviews PEPA and presents some definitions which will be used later in the paper. For a more comprehensive introduction to PEPA the reader is referred to [8].

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. We assume that there is a countable set of components, which we denote $\mathcal{C}$, and a countable set, $\mathcal{A}$, of all possible action types. We denote by $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$, the set of activities, where $\mathbb{R}^+$ is the set of positive real numbers together with the symbol $\top$.

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. These combinators are presented formally in [8]. Briefly the combinators are:

- **Prefix:** $\boldsymbol{(\alpha, r).P}$ designated first activity;
- **Choice:** $\boldsymbol{P + Q}$ alternative behaviours;
- **Cooperation:** $\boldsymbol{P \underset{L}{\bowtie} Q}$ concurrent and cooperative behaviour; we use $P \parallel Q$ to denote $P \underset{\emptyset}{\bowtie} Q$;
- **Hiding:** $\boldsymbol{P/L}$ action type abstraction;
- **Constant:** $\boldsymbol{A \stackrel{\text{def}}{=} P}$ naming of behaviour.

A *cyclic component* is one in which behaviour may always be repeated—however the model evolves from this component it will always eventually return to this component and this set of behaviours. In particular for every choice, whichever alternative is chosen the model must eventually return to the point where the choice can be made again, possibly with a different outcome. Since syntactic terms are associated with states, if we consider the layering imposed on a component by cooperation combinators, choice combinators may only be introduced at the lowest level of a cyclic component. In other words, a component which involves a choice combinator may subsequently be used in a cooperation, but a component involving a cooperation may not be subsequently used in a choice.

This leads us to formally define the syntax of PEPA expressions in terms of *sequential components* S and *model components* P:

$$\begin{aligned} P &::= S \mid P \underset{L}{\bowtie} P \mid P/L \\ S &::= (\alpha, r).S \mid S + S \mid A \end{aligned}$$

For the remainder of the paper we will assume that all the models which we consider are cyclic.

The semantics of PEPA, presented in the structured operational semantics style, are given in [8]. The underlying transition system also characterises the Markov process represented by the model. PEPA generates 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 a PEPA model as it evolves are the syntactic terms, or *derivatives*, which the model will go through. The relation $P \xrightarrow{(\alpha,r)} P'$ represents the transition from state P to state P' via an activity of type α at rate r. If $P \xrightarrow{(\alpha,r)} P'$ then P' is a one-step derivative of P.

The *derivative set* of a PEPA component C is the set of components which capture all the reachable states of the system.

Definition 3.1 (Derivative Set) *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 $C_j \in ds(C)$.*

We shall also refer to the *relative derivative set* of a component C with respect to C'.

Definition 3.2 (Relative Derivative Set) *The* relative derivative set *of a PEPA component C with respect to its derivative C' is denoted $ds_{C'}(C)$ and is defined as the smallest set of components such that*

- *if $C \stackrel{\text{def}}{=} C_0$ and $C_0 \not\equiv C'$ then $C_0 \in ds_{C'}(C)$;*
- *if $C_i \in ds_{C'}(C)$ and C' is not a one-step derivative of C_i, if there exists $a \in \mathcal{A}ct(C_i)$ such that $C_i \xrightarrow{a} C_j$ and $C_j \not\equiv C'$ then $C_j \in ds_{C'}(C)$.*

When a model component is defined it consists of one or more cooperating components, and these cooperating components will be apparent in every derivative of the model. The sequential components which are involved in the model, 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 sequential components may change.

Let P be a model component comprising sequential components $S_1, S_2, \ldots S_K$. Then the *state vector* of the model component P, as derivative P_i, is the vector $(S_{1_i}, S_{2_i}, \ldots, S_{K_i})_P$ where $S_{k_i}, 1 \le k \le K$ is the current derivative of S_k in P_i. This can be regarded as analogous to the state representation of a queueing network which consists of a vector $(n_1, n_2, \ldots, n_K)$, where n_i denotes the number of customers currently at queue i.

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 (see [8] for a formal definition). 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 $\mathcal{A}ct(P)$ is defined inductively over the structure of P, as for $\mathcal{A}(P)$ (see [8] for a formal definition).

It is also sometimes necessary to refer to the complete set of action types which are used within the complete behaviour of a component C, denoted $\vec{\mathcal{A}}(C)$.

Definition 3.3 (Complete Action Type Set) *The* complete action type set *of a component C is*

$$\vec{\mathcal{A}}(C) = \bigcup_{C' \in ds(C)} \mathcal{A}(C').$$

4 A class of reversible PEPA components

Clearly it is possible to apply Kolmogorov's criteria (2.2) directly to the underlying Markov process in order to find a reversible PEPA process (as in [1]). However, such an approach necessitates an exploration of the entire state space of the underlying Markov process and makes little use of the properties of the process algebra. It is desirable, therefore, to be able to identify reversible structure in the process algebra and to analyse these structures without having to explore the state space of the underlying Markov process. A similar idea was explored in [5] in looking at quasi-reversible structures and we proceed here with a similar argument.

Initially we aim to define a class of PEPA components which are reversible when considered in isolation. 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. A 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 (Reverse Pair) *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 $r, s \in \mathbb{R}^+$.*

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 $Q_i, i \geq 1$.

$$Q_0 \stackrel{\text{def}}{=} (arrive, \lambda).Q_1$$

$$Q_i \stackrel{\text{def}}{=} (arrive, \lambda).Q_{i+1} + (serve, \mu).Q_{i-1} \quad i \geq 1$$

When the rates of activities are unimportant we will sometimes talk about an activity a and the reverse activity $-a$.

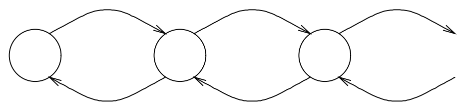
We can now define an *input-output component* related to the notion of input-output process defined in Section 2.

Definition 4.2 (Input-Output Component) *A sequential PEPA component P, with initial component P_0, is an input-output component if*

1. *for all $\alpha \in \mathcal{A}(P_0)$, such that $P_0 \xrightarrow{(\alpha,r)} P_0'$ for some r and $P_0' \not\equiv P_0$, α forms part of a reverse pair $(\alpha, -\alpha)$;*
2. *for all $P_i \in ds(P_0)$, for all $\alpha_i \in \mathcal{A}(P_i)$, such that $P_i \xrightarrow{(\alpha_i,r)} P_i'$ for some r and $P_i \not\equiv P_i'$, α_i forms part of a reverse pair $(\alpha_i, -\alpha_i)$.*

As in Section 2, this condition is not sufficient for reversibility and so we consider restricted cases of input-output components. The first two classes are restricted only on the structure of the component and are relatively simple to derive. The third class does not, strictly speaking, arise within the set of sequential components we are considering in this section. The final class imposes restrictions not only on the structure of the components, but also on the relative rates of actions enabled.

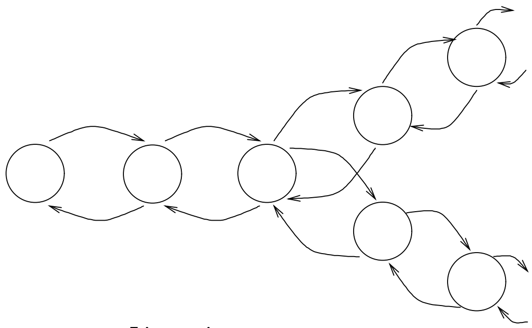
1. **Birth-death components** : In the simplest characterisation of a reversible PEPA component we only need to restrict an input-output component, P (with initial component P_0), such that every component $P_i \in ds(P_0)$ has at most two one-step derivatives (other than itself). This is easily deduced from the defining equations.
2. **Linear input-output components** : We can relax the restriction on the number of one-step derivatives a component may have as long as for every $P_i \in ds(P_0)$ we have for all Q, such that $P_i \xrightarrow{a} Q$, $ds_Q(P_i) = \{P_i\}$. This is equivalent to saying that P_i and Q only communicate via the actions a and $-a$ such that $P_i \xrightarrow{a} Q$ and $Q \xrightarrow{-a} P_i$.
3. **Parallel input-output components** : We are currently only considering sequential PEPA components; we will consider model components formed by the cooperation of two or more linear input-output components in the following section. Nevertheless, via the expansion law, it is possible for a model expressed sequentially to capture a behaviour which logically consists of the cooperation of two or more sequential components (c.f. $a.b.0 + b.a.0 \cong a.0 \parallel b.0$). Although such *embedded compositionality* may give rise to reversible components it is beyond the scope of this paper to consider a characterisation that would capture them.



Birth-death process

$$
\begin{aligned}
Gambler_0 &\stackrel{\text{def}}{=} (borrow, r).Gambler_1 \\
Gambler_n &\stackrel{\text{def}}{=} (win, w).Gambler_{n+1} \\
&\quad + \ (lose, l).Gambler_{n-1} \qquad n \geq 1
\end{aligned}
$$

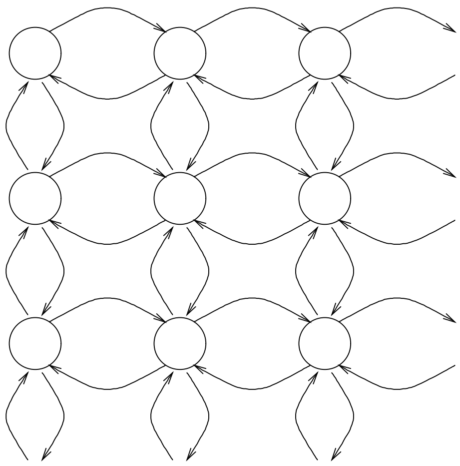
Birth-death component



Linear input-output process

$$
\begin{aligned}
Gambler_0 &\stackrel{\text{def}}{=} (borrow, r).Gambler_1 \\
Gambler_1 &\stackrel{\text{def}}{=} (win, w).Gambler_2 \ + \ (lose, l).Gambler_1 \\
Gambler_2 &\stackrel{\text{def}}{=} (win_1, w_1).Cards_3 + (win_2, w_2).Slots_3 \\
&\quad + (lose, l).Gambler_1 \\
Cards_3 &\stackrel{\text{def}}{=} (win, w).Cards_4 + (lose, l).Gambler_2 \\
Slots_3 &\stackrel{\text{def}}{=} (win, w).Slots_4 + (lose, l).Gambler_2 \\
Cards_n &\stackrel{\text{def}}{=} (win, w).Cards_{n+1} + (lose, l).Cards_{n-1} \\
Slots_n &\stackrel{\text{def}}{=} (win, w).Slots_{n+1} + (lose, l).Slots_{n-1} \\
&\qquad\qquad n \geq 4
\end{aligned}
$$

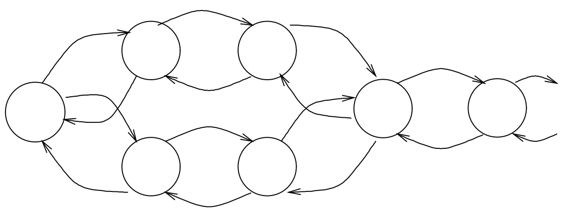
Linear input-output component



Parallel input-output process

$$
\begin{aligned}
Gambler1_0 &\stackrel{\text{def}}{=} (borrow, r).Gambler1_1 \\
Gambler1_n &\stackrel{\text{def}}{=} (win, w).Gambler1_{n+1} \\
&\qquad + \ (lose, l).Gambler1_{n-1} \\
&\qquad\qquad n \geq 1 \\
Gambler2_0 &\stackrel{\text{def}}{=} (borrow, r).Gambler2_1 \\
Gambler2_m &\stackrel{\text{def}}{=} (win, w).Gambler2_{m+1} \\
&\qquad + \ (lose, l).Gambler2_{m-1} \\
&\qquad\qquad m \geq 1 \\
Gamblers_{(n,m)} &\stackrel{\text{def}}{=} Gambler1_n \parallel Gambler2_m
\end{aligned}
$$

Parallel input-output component



Symmetrical branch-join process

$$
\begin{aligned}
Loser_0 &\stackrel{\text{def}}{=} (borrow, r_1).BJ_1 + (borrow, r_2).21_1 \\
BJ_1 &\stackrel{\text{def}}{=} (win, w).BJ_2 + (lose, l).Loser_0 \\
BJ_2 &\stackrel{\text{def}}{=} (win, w).Spin_3 \ + \ (lose, l).BJ_1 \\
21_1 &\stackrel{\text{def}}{=} (win, w).21_2 \ + \ (lose, l).Gambler_0 \\
21_2 &\stackrel{\text{def}}{=} (win, w).Spin_3 \ + \ (lose, l).21_2 \\
Spin_3 &\stackrel{\text{def}}{=} (win, w).Spin_4 + (lose, l_1).BJ_2 + (lose, l_2).21_2 \\
Spin_n &\stackrel{\text{def}}{=} (win, w).Spin_{n+1} \ + \ (lose, l).Spin_{n-1} \\
&\qquad\qquad n \geq 4
\end{aligned}
$$

Symmetrical branch-join component

Figure 1: Reversible structures

4. **Symmetrical branch-join components** : For a symmetrical branch-join component in addition to restrictions on the structure of the component we must also impose restrictions on the rates of the actions enabled. This is achieved through the notion of *symmetry* between components.

   For two components, Q and R, to be *symmetrical* they must enable the same number of current activities and moreover there must be a one-to-one correspondence such that for each $(\alpha, r) \in \mathcal{A}ct(Q)$ there is an activity $(\beta, r) \in \mathcal{A}ct(R)$. If (α, r) has a reverse activity $(-\alpha, s)$ and (β, r) has a reverse activity $(-\beta, s)$ such that $Q \xrightarrow{(\alpha,r)} Q'$, $Q' \xrightarrow{(-\alpha,s)} Q$, $R \xrightarrow{(\beta,r)} R'$ and $R' \xrightarrow{(-\beta,s)} R$ then Q and R are symmetrical if for all such pairs:

   - $Q' \equiv R'$; or
   - $Q' \equiv R$, $R' \equiv Q$ and $r = s$; or
   - Q' and R' are symmetrical.

   We can now say that a component P, with initial component P_0, is a symmetrical branch-join component, and therefore reversible, if for every $P_i \in ds(P_0)$ we have for all Q, such that $P_i \xrightarrow{(\alpha,r)} Q$, either $ds_Q(P_i) = \{P_i\}$, or Q is symmetrical to another component R such that $P_i \xrightarrow{(\beta,r)} R$.

Thus we have identified, in syntactic terms, a set of sequential PEPA components, based on the notion of input-output component, which give rise to reversible Markov processes. In the following section, taking linear input-output components as our canonical form, we will consider how such components can be combined using the PEPA combinators, whilst retaining the property of reversibility in the underlying Markov process.

5 Combining reversible components

In [1] Bhabuta et al. consider how the compositionality of a Markovian process algebra might be exploited in order to construct reversible processes from reversible sub-processes. Their argument relies on graphical analysis of the transition system and is based on particular examples. However, they do remark that the graphical forms encountered will be strongly influenced by the types of interaction offered by the process algebra and suggest that knowledge of the interactions involved might be used to simplify the graphical analysis.

In this section we take a more formal approach to the same problem. By carrying out an analysis based on the PEPA combinators we aim to establish when and how a reversible component may be constructed from known reversible subcomponents. We assume that we start with components P and Q which are known to be linear input-output components, and therefore reversible. We consider each of the PEPA operators in turn, to assess their effect on the underlying state space thereby determining whether it is possible for the resulting process to be reversible. The objective of the analysis is to establish a set of syntactic rules which will allow the modeller (and more importantly, a modelling tool) to determine whether a PEPA expression E represents a reversible Markov process if it is known that all subexpressions considered in isolation represent reversible Markov processes.

Prefix: $(\alpha, r).P$ There are two cases to consider depending on whether the syntactic form $(\alpha, r).P$ is already associated with a state in the reversible state space.

1. If $(\alpha, r).P \in ds(P)$, the prefix term is already an element of the state space. Then since the underlying state space of P is reversible then so is the state space of $(\alpha, r).P$ since the two are the same. The effect of the prefix in this case is simply to alter the initial state of the process, which is unimportant from the point of view of steady state analysis.
2. If $(\alpha, r).P \notin ds(P)$, the prefix term introduces a new state into the state space. The state associated with the syntactic term P is clearly in the current reversible state space but the new state $(\alpha, r).P$ is linked to this state only by the single arc—there is no corresponding transition rate $q(P, (\alpha, r).P)$. Thus the Kolmogorov criterion cannot be satisfied. Note that once the process has evolved by carrying out the activity (α, r) it has no way of returning to its initial state so the underlying state space is not even ergodic.

Choice: $P + Q$ Similarly to above, there are two cases to consider depending on whether the choice is already contained in the derivative sets of the reversible subcomponents P and Q.

1. If $P+Q \in ds(P)$ it follows that $Q \in ds(P)$ and since P is ergodic, $P \in ds(Q)$ and $P+Q \in ds(Q)$. By this reasoning, it must be the case that the two processes P and Q have the same state space—they give rise to the same reversible process. Since $P+Q \in ds(P)$, no additional state or transitions are added to the state space and since it satisfies the Kolmogorov criteria previously it continues to do so. As in the first case above, the effect is simply to alter the initial state of the process.
2. If $P+Q \notin ds(P)$ it follows that also $P+Q \notin ds(Q)$. Thus new elements are introduced into the state space: a state corresponding to $P+Q$ and states corresponding to any elements of $ds(Q)$ not in $ds(P)$. Consider the state corresponding to $P+Q$. Since $P+Q \notin ds(P)$ there can be no transition from any state in $ds(P)$ back to the state corresponding to $P+Q$. In particular $q(P+Q,P) > 0$ while $q(P,P+Q) = 0$. It is immediately apparent that the Kolmogorov criterion cannot be satisfied.

   The first activity to occur will move the process into the state space of either P or Q, and it will subsequently be impossible for the process to return to its initial state. Therefore, as in the second prefix case, the underlying Markov process is not even ergodic.

Cooperation: $P \underset{L}{\bowtie} Q$ The composition of the set L in relation to the action types in P and Q will dictate whether the cooperation of two linear input-output components P and Q is itself reversible. We distinguish the following distinct cases:

1. If $L = \emptyset$ then $P \underset{L}{\bowtie} Q \equiv P \parallel Q$ (syntactic equivalence); if $\vec{\mathcal{A}}(P) \cap L = \emptyset$ and $\vec{\mathcal{A}}(Q) \cap L = \emptyset$ then $P \underset{L}{\bowtie} Q = P \parallel Q$ (isomorphism). In either case, P and Q behave independently and always satisfy their own global balance equations and more particularly, their detailed balance equations. The composition will form a parallel input-output component and therefore will be reversible.
2. If $\vec{\mathcal{A}}(P) \cap L = \emptyset$ but $\vec{\mathcal{A}}(Q) \cap L \neq \emptyset$, Q will no longer be able to carry out action types which occur in $\vec{\mathcal{A}}(Q) \cap L$. This may have the effect of deadlocking Q or simply eliminating some aspects of its behaviour. In either case the effect is independent of P and we can consider a suitably constrained version of Q in isolation. We define a null component $Null$ as follows:

   $$Null \stackrel{\text{def}}{=} (\tau, r).Null$$

   and define a modified version of Q:

   $$\widehat{Q} \stackrel{\text{def}}{=} Q \underset{L}{\bowtie} Null.$$

   It is straightforward to rewrite $\widehat{Q}$ in sequential form. Note that $P \underset{L}{\bowtie} Q$ and $P \parallel \widehat{Q}$ have the same underlying state space (up to self loops). It follows that $P \underset{L}{\bowtie} Q$ will be reversible if, and only if, $\widehat{Q}$ is reversible.
3. $\vec{\mathcal{A}}(P) \cap L \neq \emptyset$ but $\vec{\mathcal{A}}(Q) \cap L = \emptyset$: as above by a symmetric argument—$P \underset{L}{\bowtie} Q$ will be reversible if, and only if, $P \underset{L}{\bowtie} Null$ is reversible.
4. Finally we consider the case $\vec{\mathcal{A}}(P) \cap L = \vec{\mathcal{A}}(Q) \cap L \neq \emptyset$, when there is genuine cooperation induced by L and no one-sided blocking. The remaining cases, such as $\vec{\mathcal{A}}(P) \cap L \subset \vec{\mathcal{A}}(Q) \cap L \neq \emptyset$, can be derived from the cases we have considered.

   Consider a sequence of states $(P_1, Q_1), \ldots, (P_{n-1}, Q_{n-1}), (P_n, Q_n), (P_1, Q_1)$ and let $\mathcal{A}ct_i$ be the multiset of activities corresponding to the state change from (P_i, Q_i) to (P_{i+1}, Q_{i+1}),

   $$\mathcal{A}ct_i = \{\!|\, (\alpha, r) \mid (\alpha, r) \in \mathcal{A}ct(P_i \underset{L}{\bowtie} Q_i) \wedge P_i \underset{L}{\bowtie} Q_i \xrightarrow{(\alpha, r)} P_{i+1} \underset{L}{\bowtie} Q_{i+1} \,|\!\}$$

   Let $\mathcal{K}(C_1, \ldots, C_n)$ denote the Kolmogorov product $q(C_1, C_2) \cdots q(C_{n-1}, C_n) q(C_n, C_1)$; similarly let $\mathcal{K}^-(C_1, \ldots C_n)$ denote the reverse product $q(C_1, C_n) q(C_n, C_{n-1}) \cdots q(C_2, C_1)$.

   First, consider sequences which involve only individual activities of P. If, for all i, $1 \leq i \leq n$, $\mathcal{A}ct_i \subseteq \mathcal{A}ct(P_i) \setminus L$ then $Q_i \equiv Q_1$ and $q((P_i, Q_i), (P_{i+1}, Q_{i+1})) = q(P_i, P_{i+1})$. Moreover it follows that

   $$\mathcal{K}((P_1, Q_1), \ldots, (P_{n-1}, Q_{n-1}), (P_n, Q_n)) = \mathcal{K}^-((P_1, Q_1), \ldots, (P_{n-1}, Q_{n-1}), (P_n, Q_n))$$

   since P is reversible and therefore satisfies the Kolmogorov criterion:

   $$\mathcal{K}(P_1, P_2 \ldots, P_{n-1}, P_n) \;=\; \mathcal{K}^-(P_1, P_2 \ldots, P_{n-1}, P_n)$$

By a similar argument, any sequence involving only individual activities of Q must also satisfy the criterion since the criterion is satisfied in Q.

Now we consider sequences which involve individual activities of both P and Q but no shared activities. In this case, at each state change only one of the processes P or Q (and their corresponding Markov processes) will change state. Thus for all i,

$$q((P_i,Q_i),(P_{i+1},Q_{i+1})) = \begin{cases} q(P_i,P_{i+1}) & \text{if the } i\text{th activity is an individual activity of } P_i \\ q(Q_i,Q_{i+1}) & \text{if the } i\text{th activity is an individual activity of } Q_i. \end{cases}$$

Furthermore, there must be a subsequence $j_1, j_2, \ldots j_{P_n}$ of $1, 2, \ldots, n$ such that the sequence of derivatives of P is $P_{j_1}, P_{j_2}, \ldots, P_{j_{n_P}}$ and $P_1 \equiv P_{j_1}$. Similarly, there must be a subsequence $k_1, k_2, \ldots, k_{Q_n}$ such that $Q_{k_1}, Q_{k_2}, \ldots, Q_{k_{n_Q}}$ is the projection of the sequence onto the Q component and $Q_1 \equiv Q_{k_1}$. Then it follows that:

$$\begin{aligned} &\mathcal{K}((P_1,Q_1),\ldots,(P_{n-1},Q_{n-1}),(P_n,Q_n)) \\ &\quad= \mathcal{K}(P_{j_1},P_{j_2},\ldots,P_{j_{n_P-1}},P_{j_{n_P}}) \times \mathcal{K}(Q_{k_1},Q_{k_2},\ldots,Q_{k_{n_Q-1}},Q_{k_{n_Q}}) \\ &\quad= \mathcal{K}^-(P_{j_1},P_{j_2},\ldots,P_{j_{n_P-1}},P_{j_{n_P}}) \times \mathcal{K}^-((Q_{k_1},Q_{k_2},\ldots,Q_{k_{n_Q-1}},Q_{k_{n_Q}}) \\ &\quad= \mathcal{K}^-((P_1,Q_1),\ldots,(P_{n-1},Q_{n-1}),(P_n,Q_n)) \end{aligned}$$

Finally, we consider the case of a sequence in which P and Q carry out at least one joint activity. Suppose that $P_i \not\equiv P_{i+1}$ and $Q_i \not\equiv Q_{i+1}$, then $(\alpha_i, r) \in \mathcal{A}ct_i$ implies that (α_i, r) must be a shared activity. There must be corresponding individual activities in P_i and Q_i such that $P_i \xrightarrow{(\alpha_i,r_P)} P_{i+1}$ and $Q_i \xrightarrow{(\alpha_i,r_Q)} Q_{i+1}$.

Moreover since P is reversible there exists (β, s_P) such that $P_{i+1} \xrightarrow{(\beta,s_P)} P_i$; similarly, since Q is reversible there exists (γ, s_Q) such that $Q_{i+1} \xrightarrow{(\gamma,s_Q)} Q_i$.

For $P \underset{L}{\bowtie} Q$ to be the reversible it must be an input-output component, i.e. there exists an activity $(-\alpha_i, r^-)$ such that $(P_{i+1},Q_{i+1}) \xrightarrow{(-\alpha_i,r^-)} (P_i,Q_i)$. Thus it follows that $\beta = -\alpha = \gamma$ and $-\alpha \in L$.

It remains to consider when the Kolmogorov criterion is satisfied. Let $\mathcal{K}(C_1,\ldots,C_n \mid C_i)$ denote the Kolmogorov product with the ith term omitted: $q(C_1,C_2)\cdots q(C_{i-1},C_i)q(C_{i+1},C_{i+2})\cdots q(C_n,C_1)$; Similarly $\mathcal{K}^-(C_1,\ldots,C_n \mid C_i)$ denote the reverse product with the term $q(C_{i+1},C_i)$ omitted.

Then we wish to show that

$$\mathcal{K}((P_1,Q_1),\ldots,(P_n,Q_n)) = \mathcal{K}^-((P_1,Q_1),\ldots,(P_n,Q_n))$$

Assuming, without loss of generality, that (P_i, Q_i) is the only state that enables a joint activity, we can project the sequence P_1, P_n onto a sequence of P derivatives $P_{j_1}, \ldots, P_{j_{n_P}}$ as previously; similarly we can form the sequence of Q derivatives $Q_{k_1}, \ldots, Q_{k_{n_Q}}$.

$$\begin{aligned} &\mathcal{K}((P_1,Q_1),\ldots,(P_n,Q_n)) \\ =\ &\mathcal{K}((P_1,Q_1),\ldots(P_n,Q_n) \mid (P_i,Q_i)) \times r \\ =\ &\mathcal{K}(P_{j_1},\ldots,P_{j_{n_P}} \mid P_i) \times \mathcal{K}(Q_{k_1},\ldots,Q_{k_{n_Q}} \mid Q_i) \times r \end{aligned}$$

Similarly,

$$\begin{aligned} &\mathcal{K}^-((P_1,Q_1),\ldots,(P_n,Q_n)) \\ =\ &\mathcal{K}^-((P_1,Q_1),\ldots(P_n,Q_n) \mid (P_i,Q_i)) \times r^- \\ =\ &\mathcal{K}^-(P_{j_1},\ldots,P_{j_{n_P}} \mid P_i) \times \mathcal{K}^-(Q_{k_1},\ldots,Q_{k_{n_Q}} \mid Q_i) \times r^- \end{aligned}$$

Since P is reversible it must satisfy the Kolmogorov criterion, hence

$$\mathcal{K}(P_{j_1},\ldots,P_{j_{n_P}}) = \mathcal{K}(P_{j_1},\ldots,P_{j_{n_P}}|P_i) \times r_P = \mathcal{K}^-(P_{j_1},\ldots,P_{j_{n_P}}|P_{i+1}) \times s_P = \mathcal{K}^-(P_{j_1},\ldots,P_{j_{n_P}})$$

and similarly for Q.

It follows that $\dfrac{r}{r^-} = \dfrac{r_P \times r_Q}{s_P \times s_Q}$ where, by definition $r = \dfrac{r_P}{r_{\alpha_i}(P_i)} \dfrac{r_Q}{r_{\alpha_i}(Q_i)} \min(r_{\alpha_i}(P_i), r_{\alpha_i}(Q_i))$ and $r^- = \dfrac{s_P}{r_{-\alpha_i}(P_{i+1})} \dfrac{s_Q}{r_{-\alpha_i}(Q_{i+1})} \min(r_{-\alpha_i}(P_{i+1}), r_{-\alpha_i}(Q_{i+1}))$ From this we can deduce that the Kolmogorov criterion will be satisfied if and only if the rates of the individual activities satisfy the following condition:

$$\frac{r_{\alpha_i}(P_i) \times r_{\alpha_i}(Q_i)}{r_{-\alpha_i}(P_{i+1}) \times r_{-\alpha_i}(Q_{i+1})} = \frac{\min(r_{\alpha_i}(P_i), r_\alpha(Q_i))}{\min(r_{-\alpha}(P_{i+1}), r_{-\alpha}(Q_{i+1}))}$$

Hiding: P/L The effect of the hiding operator is purely functional in that the timing behaviour of the component is unaffected—only the action types become hidden, the transition rates remain the same. Moreover the structure of the underlying state space does not change. Thus, if P satisfies the Kolmogorov criteria, P/L will also satisfy the criteria and will, therefore, be reversible.

Limitations It would be overly optimistic to hope that we would be able to characterise all PEPA models which will give rise to reversible Markov processes, and that has not been our objective. For example, consider the case of components P and Q which are not, when considered in isolation, individually reversible but which form a reversible component when composed in such a way that the non-reversible elements of their behaviour become blocked.

$$
\begin{array}{rcll}
P_0 & \stackrel{\text{def}}{=} & (in_P, r).P_1 + (a, r).P'_1 & \\
P_i & \stackrel{\text{def}}{=} & (in_P, r).P_{i+1} + (out_P, s).P_{i-1} & 1 \leq i \leq N-1 \\
P_N & \stackrel{\text{def}}{=} & (out_P, s).P_{N-1} & \\
P'_i & \stackrel{\text{def}}{=} & (a, r).P'_{i+1} & 1 \leq i \leq M-1 \\
P'_M & \stackrel{\text{def}}{=} & (a, r).P_0 & \\
\\
Q_0 & \stackrel{\text{def}}{=} & (in_Q, r).Q_1 + (b, r).Q'_1 & \\
Q_i & \stackrel{\text{def}}{=} & (in_Q, r).Q_{i+1} + (out_Q, s).Q_{i-1} & 1 \leq i \leq N-1 \\
Q_N & \stackrel{\text{def}}{=} & (out_Q, s).Q_{N-1} & \\
Q'_i & \stackrel{\text{def}}{=} & (b, r).Q'_{i+1} & 1 \leq i \leq M-1 \\
Q'_M & \stackrel{\text{def}}{=} & (b, r).Q_0 & \\
\\
Sys & \stackrel{\text{def}}{=} & P \underset{\{a,b\}}{\bowtie} Q &
\end{array}
$$

Here, because the action types a and b appear in the cooperation set but are not matched by the other component the corresponding activities become blocked in the cooperation and the behaviour of the component Sys is that of two independent birth-death processes. However since P and Q are not individually reversible our detection procedure would fail to identify Sys as reversible. Note though, that the presence of redundant behaviour within defining equations, i.e. the (a, r) activities in P and the (b, r) activities in Q, could be detected by the PEPA Workbench and notified to the user as a possible error.

6 Examples

In this section we present two models which exemplify how reversible structures can be recognised using our criteria. In each case the models consist of components which may be readily recognised as reversible in isolation, and which have been combined to form the model in a way that satisfies the conditions outlined in the previous section.

6.1 A drinking gambler

This example consists of two components, one representing the number of betting chips the gambler has and the other representing the gambler's behaviour. The gambler has two possible behaviours, betting and drinking.

First we consider the component $Money$ with initial state $Money_0$:

$$
\begin{array}{rcll}
Money_0 & \stackrel{\text{def}}{=} & (borrow, \gamma).Money_1 & \\
Money_j & \stackrel{\text{def}}{=} & (win, \top).Money_{j+1} + (lose, \top).Money_{j-1} & 1 \leq j \leq n-1 \\
Money_n & \stackrel{\text{def}}{=} & (lose, \top).Money_{n-1} &
\end{array}
$$

Clearly every action enabled in the components $Money_0$, $Money_j$ $(1 \leq j \leq n-1)$ and $Money_n$ forms part of a reverse pair, i.e. $(win, lose)$ is a reverse pair and so is $(borrow, lose)$. Hence this component is an input-output component. Also, no component $Money_i$ has more than two one-step derivatives, so this is also a birth-death component. Similarly the gambler's behaviour, *Gambler*, is also a birth-death component as the actions *startDrink* and *endDrink* form a reverse pair.

$$\begin{aligned}
Gambler_{bet} &\stackrel{\text{def}}{=} (win, \lambda).Gambler_{bet} + (lose, \mu).Gambler_{bet} \\
&\quad + (startDrink, \alpha).Gambler_{bet} \\
Gambler_{drink} &\stackrel{\text{def}}{=} (endDrink, \beta).Gambler_{bet}
\end{aligned}$$

Hence, the model component,

$$Money_0 \underset{(win,lose)}{\bowtie} Gambler_{bet}$$

is the cooperation of two birth-death components. We have the case that $\vec{\mathcal{A}}(P) \cap L = \vec{\mathcal{A}}(Q) \cap L \neq \emptyset$. All actions in the cooperation set L form a reverse pair with another action in L, i.e. $(win, lose)$ is a reverse pair. Moreover, in every component of *Gambler* when one half of this reverse pair is enabled, so is the other. Hence no actions which restrict the sequence of reversible transitions are blocked in either component and so the cooperation preserves reversibility. This gives us a product form for this model based on the two components, *Gambler* and *Money*.

If we were to redefine $Gambler_{drink}$ as,

$$\begin{aligned}
Gambler_{drink} &\stackrel{\text{def}}{=} (win, \lambda).Gambler_{drink} + (lose, \mu).Gambler_{drink} \\
&\quad + (endDrink, \beta).Gambler_{bet}
\end{aligned}$$

i.e. allow the gambler to drink and bet simultaneously without any change in rates, then the cooperation is simply the parallel combination of two reversible components, which is itself reversible. However, if the rates of *win* or *lose* are altered in anyway by the gambler's drinking, e.g.

$$\begin{aligned}
Gambler_{drink} &\stackrel{\text{def}}{=} (win, \delta).Gambler_{drink} + (lose, \theta).Gambler_{drink} \\
&\quad + (endDrink, \beta).Gambler_{bet}
\end{aligned}$$

where $\delta \neq \lambda$ or $\theta \neq \mu$, then the cooperation will not preserve reversibility and the product form result does not hold.

This example is analogous to an M/M/1/n queue with server vacations.

6.2 Two gamblers toss a coin

We now consider the situation in which two gamblers bet on the outcome of a toss of a coin. It can be represented by the following model,

$$\begin{aligned}
GamblerA_0 &\stackrel{\text{def}}{=} (borrow_B, \top).GamblerA_1 \\
GamblerA_i &\stackrel{\text{def}}{=} (winA, \alpha).GamblerA_{i+1} + (winB, \top).GamblerA_{i-1} && 1 \leq i \leq n-1 \\
GamblerA_n &\stackrel{\text{def}}{=} (borrow_A, \lambda).GamblerA_{n-1}
\end{aligned}$$

$$\begin{aligned}
GamblerB_0 &\stackrel{\text{def}}{=} (borrow_A, \top).GamblerB_1 \\
GamblerB_j &\stackrel{\text{def}}{=} (winA, \top).GamblerB_{j-1} + (winB, \beta).GamblerB_{j+1} && 1 \leq j \leq n-1 \\
GamblerB_n &\stackrel{\text{def}}{=} (borrow_B, \mu).GamblerB_{n-1}
\end{aligned}$$

$$GamblerA_k \underset{\substack{\{winA, borrowA \\ winB, borrowB\}}}{\bowtie} GamblerB_{n-k} \qquad \text{where } k \text{ is a constant such that } 0 \leq k \leq n.$$

The behaviour of this model is such that there is a constant amount of money distributed between the two gamblers. If one gambler has all the money then he lends one unit to the other gambler and they play on. It is clear once again that the two components are birth-death components, since all actions form reverse pairs and no component has more than two derivatives.

As above we have the case that $\vec{\mathcal{A}}(P) \cap L = \vec{\mathcal{A}}(Q) \cap L \neq \emptyset$. However, the dependencies between these components are much more restrictive than in the previous example. All the actions of *GamblerA* are dependent on same actions being performed by *GamblerB* and vice versa.

Consider the state of the system for $GamblerA_i$ and $GamblerB_j$ where $2 \leq i, j \leq n-2$. In this case, every action that can occur (*winA* or *winB*) forms part of a reverse pair with another action that can occur, i.e. only reversible transitions are possible. Now consider the case when $i = 1$. Observe that an action of type *winA* will result in a state $(GamblerA_2, GamblerB_{j-1})$ and that this action can be reversed if and only if $GamblerB_{j-1}$ enables a $winB$ action, i.e. if $2 \leq j \leq n$. Similarly, the action $winB$ will result in a state $(GamblerA_0, GamblerB_{j+1})$ and that this action can be reversed if and only if $GamberB_{j+1}$ enables a $borrow_B$ action, i.e. if $j+1=n$. Furthermore note that if the reverse actions are not enabled the model is deadlocked as no actions are enabled.

By symmetry we can make a similar argument for the case that $j=1$. Thus we can see that for this model the condition for reversibility is the condition that the model has no deadlocks. The initial state of the system, $(GamblerA_k, GamblerB_{n-k})$ where $0 \leq k \leq n$, determines that it is deadlock free and hence reversible.

7 Conclusions and further work

We have presented here a characterisation of a class of reversible components in PEPA; we have also shown how these components can be combined to preserve reversibility. Although developed here in the context of PEPA, our characterisation of sequential components which generate reversible structures is equally applicable to the other SPA languages. The criteria for when parallel compositions of reversible components will be reversible will need to be adapted for each language. This is because these conditions are intimately related to the rate of the shared activity and how this is defined differs in each of the languages. However, since the definitions in the other languages are simpler than in PEPA, we anticipate that the revised criteria would be straightforward to derive.

Clearly there is much more to be done in this area. Firstly, the class of reversible processes which we consider is restricted and we would like to be able to extend this. In addition we would like to be able to exploit reversibility in the automated solution of PEPA models with the PEPA Workbench [4].

Ultimately, only a small fraction of the number of real system models are likely fall within our characterised class of reversible models. The true value of the characterisation will be gained, not from this type of direct application, but when it is used as the target for model manipulations. Term re-writing procedures, based on the axioms of Markovian bisimulation, will allow the recognition of models which have the correct underlying structures but have not been expressed in the canonical reversible form. Model simplification procedures, leading to approximate solutions, will allow replacement of one model by a related one which does fall within the class of reversible models. Depending on the relationship between the replacement model and the original, the solution may not be exact but it will be much more efficient. These procedures are discussed in much more detail in [10].

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council via the COMPA project (G/L10215).

References

[1] M. Bhabuta, P.G. Harrison and K. Kanani, Detecting reversibility in Markovian Process Algebra, in: M. Merabti, M. Carew and F. Ball eds., *Performance Engineering of Computer and Telecommunications Systems*, Springer-Verlag, 1995.

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

[3] P. Buchholz, Compositional Analysis of a Markovian Process Algebra, in: U. Herzog and M. Rettelbach eds., *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Regensburg, July, 1994.

[4] G. Clark, S. Gilmore, J. Hillston and N. Thomas, Experiences with the PEPA performance modelling tools, in: R. Pooley and N. Thomas eds., *Proceedings of the 14th UK Performance Engineering Workshop*, Dept of Computer Science, The University of Edinburgh, July 1998.

[5] P.G. Harrison and J. Hillston, Exploiting Quasi-reversible Structures in Markovian Process Algebra Models, *The Computer Journal*, **38**(7), pp. 510–520, 1995.

[6] P.G. Harrison and N.M. Patel, *Performance Modelling of Communication Networks and Computer Architectures*, Addison-Wesley, 1993.

[7] J. Hillston, The Nature of Synchronisation, in: U. Herzog and M. Rettelbach eds., *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Regensburg, July, 1994.

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

[9] J. Hillston, A Class of PEPA Models Exhibiting Product Form Solution over Submodels, Technical Report LFCS-98-382, Department of Computer Science, University of Edinburgh, 1998.

[10] J. Hillston, Exploiting Structure in Solution: Decomposing Composed Models, in: *Proceedings of the 6th Process Algebra and Performance Modelling Workshop*, Nice, 1998.

[11] J. Hillston and V. Mertsiotakis, A Simple Time Scale Decomposition Technique for Stochastic Process Algebras, *The Computer Journal*, **38**(7), pp. 566–577, 1995.

[12] J. Hillston and N. Thomas, Product Form Solution for a class of PEPA Models, *to appear in IPDS'98*, 1998.

[13] F.P. Kelly, *Reversibility and Stochastic Networks*, Wiley, 1979.

[14] A.A. Lazar and T.G. Robertazzi, Markovian Petri net protocols with product form solution, *Performance Evaluation*, **12**, pp. 67–77, 1991.

[15] V. Mertsiotakis, Time Scale Decomposition of Stochastic Process Algebra Models, in: *Proceedings of the 5th Process Algebra and Performance Modelling Workshop*, University of Twente, 1997.

[16] V. Mertsiotakis and M. Silva, A Throughput Approximation Algorithm for Decision Free Processes, in: *Proceedings of the 4th Process Algebra and Performance Modelling Workshop*, University of Torino, 1996.

[17] M. Rettelbach and M. Siegle, Compositional Minimal Semantics for the Stochastic Process Algebra TIPP, in: U. Herzog and M. Rettelbach eds., *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Regensburg, July, 1994.

[18] M. Sereno, Towards a product form solution for stochastic process algebras, *The Computer Journal*, **38**(7), pp. 622–632, 1995.

[19] P. Whittle, *Systems in Stochastic Equilibrium*, Wiley, 1986.