PRODUCT FORM OVER ON-OFF COMPONENTS IN PEPA

NIGEL THOMAS

Research Institute in Software Evolution
University of Durham, UK. Nigel.Thomas@durham.ac.uk

Abstract: In the study of stochastic process algebra it is necessary to consider not only how systems are to be specified, but also how complex systems can be simplified and solved efficiently. In this paper a relationship between the behaviour of stochastic systems and a product form solution over components is explored. A number of characterisations of increasing complexity are derived which extend the class of model subject to product form solution that have been defined for Markovian process algebra.

Keywords: Product form, Markovian process algebra, PEPA, decomposition

1. INTRODUCTION

In recent years some effort has been made to identify efficient methods for analysing and solving stochastic process algebra models by decomposition (see [Hillston, 2001]). Such solutions are derived on the basis that the components in the model are statistically independent in their steady state behaviour and so the steady state solutions for components may be found in isolation without the need to generate the entire state space of the model. Clearly product form solutions are an extremely efficient mechanism in deriving important numerical solutions.

The aim of this paper is to address the issue of how the behaviour of the model may be used to directly show product form results in general models without relying on additional insight from the modeller. The product form solutions that are derived here are related to the class of model previously defined by Boucherie [Boucherie, 1994]. This class of model gives a product form over components interacting only through a resource, to which exclusive access is granted to one component at a time. No other interaction is possible between components and without the resource a component may only carry out internal actions.

The paper begins by re-introducing Hillston's Markovian process algebra, PEPA [Hillston, 1996], together with the set of concepts required to describe features of a model and then briefly discusses the notion of behavioural independence and control. In Section 3 the exploitation of behavioural independence is made in relation to simple product form decomposition. In Section 4 this is developed to consider more complex classes of model. Finally some conclusions and future work directions are presented.

2. PEPA

A formal presentation of PEPA is given in [Hillston, 1996], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that are negative exponentially distributed at given rates. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the *type* of the activity, α, and the *rate* of the associated negative exponential distribution, r. This rate may be any positive real number, or given as *unspecified* using the symbol $\top$. The syntax for describing components is given as:

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

The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P. The component $P+Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete. The component P/L behaves exactly like P except that

Proceedings 17th European Simulation Multiconference
(c) SCS Europe BVBA, 2003 ISBN 3-936150-25-7

the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ. Concurrent components can be synchronised, $P \underset{L}{\bowtie} Q$, such that activities in the *cooperation set* L involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to the activities of that type. The parallel combinator $\|$ is used as shorthand to denote synchronisation with no shared activities, i.e. $P\|Q \equiv P \underset{\emptyset}{\bowtie} Q$. $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P. A small number of addition definitions are required.

DEFINITION 1: Fertile action. An action γ is said to be fertile in derivative P_i if $P_i \xrightarrow{\gamma} P_j$ and $i \neq j$.

DEFINITION 2: Current fertile action type set. The current fertile action type set of P, denoted $\mathcal{A}_f(P)$, is the set of all action types of actions that are fertile in the current derivative of P.

DEFINITION 3: Complete fertile action type set. The complete fertile action type set of P, denoted $\vec{\mathcal{A}}_f(P)$, is the set of all action types of actions that are fertile in at least one derivative of P.

3. BEHAVIOURAL INDEPENDENCE

Put simply the notion of *behavioural independence* is simply that a component in a model behaves identically regardless of the current behaviour of the other components in the model.

DEFINITION 4: Behavioural Independence. The component P is said to be behaviourally independent in the model $P \underset{L}{\bowtie} Q$ if for every $P_i \in ds(P)$ $\forall\ Q_j, Q_k \in ds(Q)\ s.t.\ (P_i \underset{L}{\bowtie} Q_j), (P_i \underset{L}{\bowtie} Q_k) \in ds(P \underset{L}{\bowtie} Q)$

$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$
$$=$$
$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$

Obviously the trivial case for behavioural independence is where there are no shared actions, i.e. $P\|Q$, however this is not the only case where components may be considered to be behaviourally independent. Furthermore, the fact that no actions are shared between two components does not mean they will always be behaviourally independent in the presence of other components. For example, in $(P\|Q) \underset{L}{\bowtie} R$ the interaction between P and R may influence the interaction between Q and R, causing P and Q to be behaviourally *dependent*. If a component is not behaviourally independent then it must be dependent on some other component to perform one of more actions during its evolution. This dependence is referred to by saying that component P *controls* component Q over actions $K \subset L$ in $P \underset{L}{\bowtie} Q$ if the rate at which an action of type $k \in K$ can happen in $Q_i \in dsQ$ depends on the current derivative of P. Clearly, if P controls Q over K then Q cannot be behaviourally independent, but the independence, or otherwise, of P is not known by this statement.

3.1 Exploiting behavioural independence

It is clear that if a component is behaviourally independent (even it is also controlling) then it may be studied in isolation without affecting its behaviour, subject to the rates of shared actions being set. If a shared action is not enabled by the partner in the cooperation then the rate of that action will be zero when that component is considered in isolation, otherwise the rate of the shared action will be determined by the rates specified in each participating component. To illustrate such a situation, consider a number of queues in sequence. If all the queues are basic $M/M/1/\infty$ then the system is clearly a simple Jackson network and has a product form solution. However, even if this is not the case then the first queue will still be behaviourally independent (unless there is blocking at the server) and so may be studied in isolation.

As well as being used to identify independent behaviour leading to decomposition, behavioural independence and control can also be used to identify cases where product form solutions exist. Such a case is the queueing model with breakdowns illustrated below.

$$\begin{aligned}
Queue_0 &\stackrel{def}{=} (arrival, \top).Queue_1 \\
Queue_i &\stackrel{def}{=} (arrival, \top).Queue_{i+1} \\
&\quad +(service, \top).Queue_{i-1} \\
&\quad ,\ 1 \leq j \leq N-1 \\
Queue_N &\stackrel{def}{=} (service, \top).Queue_{N-1} \\
Server_{on} &\stackrel{def}{=} (fail, \xi).Server_{off} \\
&\quad +(arrival, \lambda).Server_{on} \\
&\quad +(service, \mu).Server_{on}
\end{aligned}$$

$$Server_{off} \stackrel{def}{=} (repair, \eta).Server_{on}$$

$$Queue_0 \underset{\{service, arrival\}}{\bowtie} Server_{on}$$

It is clear that the *Server* component is behaviourally independent in this model as neither of the shared actions affects its evolution. Similarly it is clear that the *Server* component controls the *Queue* component over the actions *service* and *arrival*. A number of other important factors are also apparent: all the actions of *Queue* are shared actions, all shared actions are enabled in $Server_{on}$, no shared actions are enabled in $Server_{off}$, the action *fail* does not alter the derivative of *Queue*. These six factors mean that a product form solution exists over the *Server* and *Queue* components such that the joint steady state probabilities are given as $\pi_{(Server_j, Queue_i)} = \pi_{Server_j}.\pi_{Queue_i}$ where $j \in \{on, off\}$ and $0 \leq i \leq N$.

The model illustrated in here is reversible and it is possible to derive a product form solution using the characterisation derived in [Hillston and Thomas, 1998]. In fact an example with the same structure, referred to as *the drinking gambler* appeared in [Hillston and Thomas, 1998]. That approach required the identification of reversible components and the application of restrictions on the cooperation between them. This requires a detailed study of both the components and the interface, whereas the approach described here only requires a simple inspection of the components, only adherence to the five criteria.

1. Component, A, of a pair $A \underset{L}{\bowtie} B$ is behaviourally independent.
2. Component, B, is controlled by A over all the actions in the cooperation set, $\mathcal{K}(B) = L$.
3. The complete action type set of A, $\vec{\mathcal{A}}(B)$ is contained within its interface, $\vec{\mathcal{A}}(B) = L$.
4. All actions in the cooperation set, L, are enabled in exactly one derivative of A, A_i.
5. No actions in the cooperation set, L, are enabled in any other derivative of A.
6. Any action α such that $A_i \xrightarrow{\alpha} A_j$, $A_i \neq A_j$ is not fertile in B.

This product form solution relies on the fact that component A turns the interaction in the model off and on; and when it turns back on the system returns to exactly the same global state as it was before it turned off. Hence, as long as these 5 stated conditions are not broken then the model illustrated above can be easily adapted to incorporate additional (non-reversible) features, such as batch service, without compromising the product form solution.

$$\begin{aligned}
Queue_0 &\stackrel{def}{=} (arrival, \top).Queue_1 \\
Queue_i &\stackrel{def}{=} (arrival, \top).Queue_{i+1} \\
&\quad +(service, \top).Queue_0 \\
&\quad , \ 1 \leq j \leq N-1 \\
Queue_N &\stackrel{def}{=} (service, \top).Queue_0 \\
Server_{on} &\stackrel{def}{=} (fail, \xi).Server_{off} \\
&\quad +(arrival, \lambda).Server_{on} \\
&\quad +(service, \mu).Server_{on} \\
Server_{off} &\stackrel{def}{=} (repair1, \eta_1).Server_{standby} \\
Server_{standby} &\stackrel{def}{=} (repair2, \eta_2).Server_{on}
\end{aligned}$$

$$Queue_0 \underset{\{service, arrival\}}{\bowtie} Server_{on}$$

4. PARTIAL BEHAVIOURAL INDEPENDENCE

The simple product form developed in Section 3 can be extended by considering parts of components as behaviourally independent and parts which exert control. This may be achieved by observing that component P controls Q over the set of actions K, but that the actions in K are not affected by changes in derivative within a subset of P.

DEFINITION 5: Partial Behavioural Independence The component P is said to have partial behavioural independence in the model $P \underset{L}{\bowtie} Q$ with respect to the subset $\mathcal{D}(Q) \subset ds(Q)$, if for every $P_i \in ds(P)$ $\forall \ Q_j, Q_k \in \mathcal{D}(Q) \ s.t. \ (P_i \underset{L}{\bowtie} Q_j), (P_i \underset{L}{\bowtie} Q_k) \in ds(P \underset{L}{\bowtie} Q)$

$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$

$$=$$

$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$

This definition states that the P will behave identically as long as Q behaves as some derivative $Q_i \in \mathcal{D}(Q)$. Clearly there may be many such subsets for any given component. In product form solution introduced in Section 3.1, the component A has two subsets; one subset consists of the single derivative where all actions

in the cooperation set, L, are enabled, and the other subset consists off all the other derivatives where no actions in the cooperation set are enabled. The single "on" behaviour restriction from A can be relaxed as long as the result of returning to "on" always returns to exactly the same derivative of A as immediately before turning "off".

1. Component, A, of a pair $A \underset{L}{\bowtie} B$ is behaviourally independent.
2. Component, B, is controlled by A over all the actions in the cooperation set, $\mathcal{K}(B) = L$.
3. The complete action type set of B, $\vec{\mathcal{A}}(B)$ is contained within its interface, $\vec{\mathcal{A}}(B) = L$.
4. It is possible to divide the derivatives of A into N subsets $\mathcal{D}_1(A), \ldots, \mathcal{D}_N(A)$, $N \geq 2$, such that $\bigcup_{i=1}^{N} \mathcal{D}_i(A) = ds(A)$, $\mathcal{D}_i(A) \cap \mathcal{D}_j(A) = \emptyset$, $i \neq j$, and B has partial behavioural independence in $A \underset{L}{\bowtie} B$ with respect to $\mathcal{D}_i(A)$, $i = 1, \ldots, N$.
5. All actions in the cooperation set, L, are enabled in all derivatives in $\mathcal{D}_1(A)$.
6. No actions in the cooperation set, L, are enabled in any derivative in $\mathcal{D}_i(A)$, $i \geq 2$.
7. For any $l \geq 2$, there exists at most one derivative, $A_i \in \mathcal{D}_1(A)$ such that $A_i \longrightarrow A_j$ and $A_k \longrightarrow A_i$ where $A_j, A_k \in \mathcal{D}_l(A)$.
8. For any $i \neq j$, ≥ 2 there are no derivatives $A_k \in \mathcal{D}_i(A)$ and $A_l \in \mathcal{D}_j(A)$ such that $A_k \longrightarrow A_l$.
9. Any action α such that $A_i \xrightarrow{\alpha} A_j$, $A_i \in \mathcal{D}_1(A)$, $A_j \notin \mathcal{D}_1(A)$, α is not fertile in B.

This set of rules allows for multiple "on" behaviours and multiple "off" behaviours, by partitioning the "off" behaviours into distinct subsets that do not have any single actions linking them. However, this definition still imposes the restriction the "on" and "off" behaviours are controlled by a single behaviourally independent component. It is possible to relax even this restriction, however, this can be done only if the actions in the "on" subset, $\mathcal{D}_1(A)$ are restricted to the actions in the cooperation set. This further restriction would mean that two components A and C, could both control B over L, and also that A controls C over L and C controls A over L. In fact it it not necessary for the component B to be constrained by its interface, and it can in fact be an on-off component in the same way as A.

DEFINITION 6: Restricted Partial Behavioural Independence The component P is said to have restricted partial behavioural independence in the model $P \underset{L}{\bowtie} Q$ with respect to the subsets $\mathcal{D}(Q) \subset ds(Q)$ and $\mathcal{D}(P) \subset ds(P)$, if for every $P_i \in \mathcal{D}(P)$
$\forall\ Q_j, Q_k \in \mathcal{D}(Q)\ s.t.\ (P_i \underset{L}{\bowtie} Q_j), (P_i \underset{L}{\bowtie} Q_k) \in ds(P \underset{L}{\bowtie} Q)$

$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$
$$=$$
$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$

Thus it is possible to derive a product form solution in a model without a resource component subject to the following conditions.

1. Two components, A and B in $A \underset{L}{\bowtie} B$ are controlled and controlling over all the actions in the cooperation set, $\mathcal{K}(B) = \mathcal{K}(A) = L$.
2. It is possible to divide the derivatives of A into N distinct subsets $\mathcal{D}_1(A), \ldots, \mathcal{D}_N(A)$, $N \geq 2$, and the derivatives of B into M subsets $\mathcal{D}_1(B), \ldots, \mathcal{D}_M(B)$, $M \geq 2$ such that B has restricted partial behavioural independence in $A \underset{L}{\bowtie} B$ with respect to $\mathcal{D}_i(A)$ and $\mathcal{D}_1(B)$, $i = 1, \ldots, N$ and A has restricted partial behavioural independence in $A \underset{L}{\bowtie} B$ with respect to $\mathcal{D}_j(B)$ and $\mathcal{D}_1(A)$, $j = 1, \ldots, M$.
3. All actions in the cooperation set, L, are enabled in all derivatives in $\mathcal{D}_1(A)$ and all derivatives in $\mathcal{D}_1(B)$.
4. No actions in the cooperation set, L, are enabled in any derivative in $\mathcal{D}_i(A)$, $i \geq 2$, or any derivative in $\mathcal{D}_j(B)$, $j \geq 2$.
5. The current action type set of all $A_i \in \mathcal{D}_1(A)$, $\mathcal{A}(A_i)$, is contained within its interface; $\mathcal{A}(A_i) \subset L, \forall A_i \in \mathcal{D}_1(A)$.
6. The current action type set of all $B_j \in \mathcal{D}_1(B)$, $\mathcal{A}(B_j)$, is contained within its interface; $\mathcal{A}(B_j) \subset L, \forall B_j \in \mathcal{D}_1(B)$.
7. For any $l \geq 2$, there exists at most one derivative, $A_i \in \mathcal{D}_1(A)$ such that $A_i \longrightarrow A_j$ and $A_k \longrightarrow A_i$ where $A_j, A_k \in \mathcal{D}_l(A)$.
8. For any $i \neq j$, ≥ 2 there are no derivatives $A_k \in \mathcal{D}_i(A)$ and $A_l \in \mathcal{D}_j(A)$ such that $A_k \longrightarrow A_l$.

9. For any $l \geq 2$, there exists at most one derivative, $B_i \in \mathcal{D}_1(B)$ such that $B_i \longrightarrow B_j$ and $B_k \longrightarrow B_i$ where $B_j, B_k \in \mathcal{D}_l(B)$.

10. For any $i \neq j$, ≥ 2 there are no derivatives $B_k \in \mathcal{D}_i(B)$ and $B_l \in \mathcal{D}_j(B)$ such that $B_k \longrightarrow B_l$.

11. Any action α such that $A_i \xrightarrow{\alpha} A_j$, $A_i \in \mathcal{D}_1(A)$, $A_j \notin \mathcal{D}_1(A)$ is not fertile in B.

12. Any action β such that $B_i \xrightarrow{\alpha} B_j$, $B_i \in \mathcal{D}_1(B)$, $B_j \notin \mathcal{D}_1(B)$ is not fertile in A.

These conditions follow as a simple consequence of the earlier discussion. All interaction between A and B is restricted to the subsets $\mathcal{D}_1(A)$ and $\mathcal{D}_2(B)$. If A enters a behaviour outside $\mathcal{D}_1(A)$, then all actions of B will be blocked until A returns to $\mathcal{D}(A)$. Furthermore, if A was behaving as $A_1 \in \mathcal{D}_1(A)$ and the action α occurred causing A to behave as $A_1' \notin \mathcal{D}_1(A)$ then B will be blocked until A is once again behaving as A_1. Likewise B will block the actions of A if its behaviour leaves the subset $\mathcal{D}_1(B)$. Clearly therefore this characterisation gives rise to a separable system, but in general this does not mean a product form solution will exist. This is only guaranteed if each of the activities in L are fertile in at most one component in every derivative of $A \underset{L}{\bowtie} B$, $A' \underset{L}{\bowtie} B'$, such that $A' \in \mathcal{D}_1(A)$ and $B' \in \mathcal{D}_1(B)$ (the same condition is present in the Boucherie product form). This gives a product form solution of the following structure.

$$\pi_{(A_j, B_k)} = \frac{1}{X} \pi_{A_j} \cdot \pi_{B_k}$$

where $1/X$ is the normalising constant resulting from the fact that not all combinations of derivatives of A and B are reachable in $A \underset{L}{\bowtie} B$.

4.1 Security guards example

In this simple example there are a pair of security guards. The only stipulation is that at least one must be awake and on duty at any time. Thus, any guard may choose to go to sleep only if another of his colleagues is awake. The correctness is held by the fact that in order to make the transition from $G_x Awake$ to $G_x Asleep$, each guard must have the (passive) cooperation of his colleague. Once asleep however, the guards are incapable of communicating, and so the guard who is awake must remain so.

$$\begin{aligned}
G_A Awake &\stackrel{def}{=} (afallAsleep, r_2).G_A Asleep + \\
&\quad (bfallAsleep, \top).G_A Awake \\
G_A Asleep &\stackrel{def}{=} (wakeup, r_4).G_A Awake \\
G_B Awake &\stackrel{def}{=} (bfallAsleep, r_2).G_B Asleep \\
&\quad +(afallAsleep, \top).G_B Awake \\
G_B Asleep &\stackrel{def}{=} (wakeup, r_6).G_B Awake
\end{aligned}$$

$$G_A Awake \underset{\{afallAsleep, bfallAsleep\}}{\bowtie} G_B Awake$$

Thus the model has the excluded state of $G_A Asleep | G_B Asleep$, and a trivial product form over the remaining states, given by,

$$\pi_{(G_A Y, G_B Z)} = \frac{1}{X} \pi_{G_A Y} \cdot \pi_{G_B Z}$$

where $Y, Z = \{Awake, Asleep\}$ and $X = \pi_{G_A Awake} \cdot \pi_{G_B Awake} + \pi_{G_A Awake} \cdot \pi_{G_B Asleep} + \pi_{G_A Asleep} \cdot \pi_{G_B Awake}$. The model can be made slightly more interesting if guards go out on patrol. There are a number of possibilities in this regard.

- *Both guards patrol together at all times.*

$$\begin{aligned}
G_A Awake &\stackrel{def}{=} (goOut, r_7).G_A Patrol \\
&\quad +(afallAsleep, r_2).G_A Asleep \\
&\quad +(bfallAsleep, \top).G_A Awake \\
G_A Patrol &\stackrel{def}{=} (goBack, r_8).G_A Awake \\
G_B Awake &\stackrel{def}{=} (goOut, r_7).G_B Patrol \\
&\quad +(bfallAsleep, r_2).G_B Asleep \\
&\quad +(afallAsleep, \top).G_B Awake \\
G_B Patrol &\stackrel{def}{=} (goBack, r_8).G_B Awake
\end{aligned}$$

$$G_A Awake \underset{\substack{\{goOut, goBack, \\ afallAsleep, bfallAsleep\}}}{\bowtie} G_B Awake$$

- *One or both of the guards go out, the other must be awake.*

$$\begin{aligned}
G_A Awake &\stackrel{def}{=} (goOut, r_7).G_A Patrol \\
&\quad +(goOut, \top).G_A Awake \\
&\quad +(afallAsleep, r_2).G_A Asleep \\
&\quad +(bfallAsleep, \top).G_A Awake \\
G_A Patrol &\stackrel{def}{=} (goBack, r_8).G_A Awake \\
G_B Awake &\stackrel{def}{=} (goOut, r_9).G_B Patrol \\
&\quad +(goOut, \top).G_B Awake \\
&\quad +(bfallAsleep, r_2).G_B Asleep \\
&\quad +(afallAsleep, \top).G_B Awake \\
G_B Patrol &\stackrel{def}{=} (goBack, r_8).G_B Awake
\end{aligned}$$

$$G_A Awake \underset{\{afallAsleep, bfallAsleep, goOut\}}{\bowtie} G_B Awake$$

Here both components might be simultaneously passive on $goOut$, however this is not a fertile action in this case. This case only has a product form if $r_7 = r_9$.

- *Either guard may go out at anytime regardless of whether the other is awake or not.*

$$\begin{aligned}
G_A Awake &\stackrel{def}{=} (goOut, r_7).G_A Patrol \\
&\quad +(goOut, \top).G_A Awake \\
&\quad +(afallAsleep, r_2).G_A Asleep \\
&\quad +(bfallAsleep, \top).G_A Awake \\
G_A Patrol &\stackrel{def}{=} (goBack, r_8).G_A Awake \\
&\quad +(bfallAsleep, \top).G_A Awake \\
G_B Awake &\stackrel{def}{=} (goOut, r_9).G_B Patrol \\
&\quad +(goOut, \top).G_B Awake \\
&\quad +(bfallAsleep, r_2).G_B Asleep \\
&\quad +(afallAsleep, \top).G_B Awake \\
G_B Patrol &\stackrel{def}{=} (goBack, r_8).G_B Awake \\
&\quad +(afallAsleep, \top).G_B Awake
\end{aligned}$$

$$G_A Awake \underset{\{afallAsleep, bfallAsleep\}}{\bowtie} G_B Awake$$

This final model is not captured by the characterisation presented here, but instead belongs to a further class of model where independent actions are allowed to continue. Capturing this class of model remains ongoing work.

5. CONCLUSIONS

In this paper a discussion of the notions of behavioural independence and control has been presented in relation to product form solution. It is probable that many additional classes of product form solution will have subclasses that can be defined using behavioural independence and control, although this remains to be proved. By exploring these subclasses of solutions it will be possible to gain greater understanding of the links between different product form solutions and near and non-product form solutions and where they may overlap. The class of product form here is related to the class defined by Boucherie [Boucherie, 1994] and is thus related to an earlier characterisation in PEPA [Hillston and Thomas, 1999], although it is conceptually somewhat simpler than either of these cases. These earlier characterisations have some advantages over that presented here, however the class defined here differs from those earlier characterisations in three important ways. The equivalent of the resource component used here is not redundant. This means that queues (and other state dependent structures) can be used under a Boucherie-like framework. Furthermore, by developing the notion of *restricted partial behavioural independence* it has been possible to achieve a characterisation without an explicit resource. Given the Boucherie result it should be possible to relax the on-off behaviour such that component B in $A \underset{L}{\bowtie} B$ will still be able to perform internal actions even if A is in some derivative $A' \notin \mathcal{D}_1(A)$. This remains as future work.

REFERENCES

Boucherie R. 1994, A Characterisation of Independence for Competing Markov Chains with Applications to Stochastic Petri nets,*IEEE Transactions on Software Engineering* **20**(7).

Clark G. Gilmore S. Hillston J. and Thomas N. 1999, Experiences with the PEPA performance modelling tools, *IEE Proceedings - Software*, **146**(1).

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

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

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

Hillston J. and Thomas N. 1999, Product Form Solution for a Class of PEPA Models, *Performance Evaluation*, **35**(3-4), pp. 171-192.

Hillston J. and Thomas N. 1998, A Syntactic Analysis of Reversible PEPA Models, in: *Proceedings of the Sixth International Workshop on Process Algebra and Performance Modelling*.

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

Thomas N. 2002, Exploiting behavioural independence and control in Markovian process algebra, in: *Proceedings of the 1st Workshop on Process Algebra with Stochastic Timed Activities*, University of Edinburgh.