On the Aggregation Techniques in Stochastic Petri Nets and Stochastic Process Algebras

MARINA RIBAUDO

Dipartimento di Informatica
Università di Torino
Corso Svizzera 185
10149 Torino, Italy
e-mail: marina@di.unito.it

Stochastic Petri Nets and Stochastic Process Algebras can both be used for the qualitative and quantitative analysis of parallel and distributed systems. However, sometimes the complexity of the system can make such analysis infeasible, due to the large size of the state space underlying the model. To overcome this problem different aggregation techniques have been proposed in both formalisms. Two of them are compared in this paper.

1. INTRODUCTION

Stochastic Petri Nets (SPN) (Molloy, 1982) are a well-established formalism for the modelling and performance analysis of concurrent systems. A major drawback of SPNs is that they suffer from the so-called *state space explosion* problem. Even relatively small models can, in fact, generate a huge state space making the analysis of the system infeasible.

Stochastic Process Algebras (SPA) (Hillston, 1994; Gotz *et al.*, 1994; Bernardo *et al.*, 1994; Buchholz, 1994) provide a constructive modelling technique, suitable for functional and performance analysis, which has gained popularity in recent years. Unfortunately, also SPA models are prone to the state space explosion problem.

To cope with this problem, state space aggregation techniques have been proposed in both formalisms. In this paper we compare two aggregation techniques based on lumpability, one developed in the SPN formalism and the other developed in the SPA formalism. In particular we consider the approach for the automatic detection of symmetries in Stochastic Well Formed Nets (SWN) (Chiola *et al.*, 1993) and the aggregation technique based on an equivalence notion introduced in SPA (Hillston, 1994; Hermanns & Rettelbach, 1994).

In order to compare these two techniques a compositional SWN semantics for SPA is described, so that SPA specifications can be translated into SWN models. The notions of *horizontal* and *vertical* aggregation are then introduced to provide insight into the different level of aggregation achieved by the two techniques.

The remainder of the paper is as follows. Section 2. and Section 3. briefly introduce the SWN formalism and the equivalence notion defined in SPA for state space aggregation. The SWN semantics for SPA is described in Section 4.. The comparison between the two aggregation approaches is discussed in Section 5.. An algorithm for finding a canonical form for SPA terms, derived from the algorithm for the computation of the Canonical Symbolic Marking (Chiola *et al.*, 1990; Chiola *et al.*, 1992), is sketched in Section 5.4.. Finally, Section 6. concludes this work.

2. AGGREGATION IN SPN: STOCHASTIC WELL FORMED NETS

Stochastic Well Formed Nets (SWN) (Chiola *et al.*, 1993) are a coloured extension of SPNs that allows one to build a more compact and parametric representation of a symmetric system by *folding* similar subnets. In this way it is possible to represent very concisely systems that would have required a huge uncoloured net. When similar subnets are folded, some additional annotation is needed to distinguish tokens that end up being in the same folded place. These annotations constitute the *colour structure* of the net.

Tokens are no longer indistinguishable: each token can be regarded as an instance of a data structure whose meaning depends on the place to which the token belongs. The place *colour domain* (denoted $C(p)$) is defined as the Cartesian product of *basic colour classes*, possibly with repetitions of the same basic colour class. Each basic colour class is a finite set of basic objects and it is usually defined by enumeration of its elements (e.g. $C = \{c_1, c_2, \ldots c_n\}$). Colour classes may be partitioned into disjoint subsets called *static subclasses*.

Transitions' colour domains (denoted $C(t)$) are defined analogously to places' colour domains. Transitions can be seen as procedures with formal parameters, the parameters being determined by the corresponding domain.

The enabling check of a transition and the state

change caused by its firing depend on the *arc functions* that label the arcs connecting the transition to input, inhibitor and output places.

Arc functions are formal sums of tuples structured according to the corresponding place colour domain. If the place colour domain is the Cartesian product of k basic colour classes, then the corresponding arc function is a weighted sum of k-tuples. The j^{th} element in each k-tuple is a weighted sum of three basic functions, the *identity* function (denoted by a variable), the *successor* function ("!"), and the *synchronisation* function ("S").

The weights of the sum may be numbers or *predicates*. Predicates are logical expressions used to test either equality of pairs of basic objects, selected by some identity/successor function, or to check the membership of a selected basic object in a given static subclass.

A major interest of SWNs is that they provide a modelling framework in which the intrinsic symmetries are automatically detected and used naturally as a way for reducing the size of the underlying state space. The reduction is obtained thanks to the original concept of *symbolic marking*. Informally a symbolic marking corresponds to an equivalence class representing a set of ordinary markings characterised by a common future behaviour. These ordinary markings in fact enable the same transitions whose firings lead to new ordinary states which are still equivalent, i.e. belong to the same symbolic marking. Symbolic markings are obtained by disregarding the identities of the objects within the places of the net and considering only their number. Colour classes are partitioned into *dynamic subclasses* and the only relevant information is the cardinality of these subclasses (i.e. the number of objects they contain). This shows how many elements in the net have the same behaviour at the same time.

This type of partitioning varies from one marking to another, hence it must not be confused with the static subclass partitioning which is part of the colour class definition.

With the introduction of dynamic subclasses places no longer contain coloured tokens but symbolic tokens whose components are expressed in terms of dynamic subclasses. All the ordinary markings which can be obtained by assigning identities to the objects of the dynamic subclasses belong to the same symbolic marking.

A *symbolic enabling rule* and a *symbolic firing rule*, which operate directly on the symbolic marking representation, and an efficient algorithm for the generation of an aggregated state space called *symbolic reachability graph* (SRG) have been defined (Chiola *et al.*, 1993) and implemented (Chiola *et al.*, 1995). The SRG describes the evolution of a SWN model through a set of macro-states, the symbolic markings, that represent sets of more detailed states which are equivalent.

Several properties valid for the SRG have been introduced. For example, the equivalence between the SRG and the RG from the point of view of the reachability of the markings ensures that no information is lost by analysing the SRG instead of the RG. Formulae have been defined to compute both the number of ordinary markings belonging to the same equivalence class and the number of ordinary firings represented by each symbolic firing.

The SRG corresponds to a lumped version of the complete RG and this aggregation is reflected also at the level of the underlying Markov process. In (Chiola *et al.*, 1991) it has been proved that the SRG is isomorphic to an aggregated Markov process that can be used to compute the same performance estimates that can be computed from the general technique based on the RG, but with a lower computational cost.

We end this section with a formal definition of SWNs (Chiola *et al.*, 1993) which we will need in Section 4. when defining the SWN semantics for SPA.

DEFINITION 2.1. *A Stochastic Well Formed net is defined as* $SWN = (P, T, C, J, W^-, W^+, \Phi, \pi, M_0, \theta)$ *where*

1. *P is the finite set of places;*
2. *T is the finite set of transitions;*
3. *C is the family of basic colour classes:*
   $C = \{C_1, C_2, \ldots, C_n\}$, with $C_i \cap C_j = \emptyset$ (we denote $I = \{1, \ldots, n\}$ the ordered set of indexes);
   each class C_i is possibly partitioned into static subclasses: $C_i = \bigcup_{q=1}^{n_i} D_{i,q}$;
4. *$J : P \cup T \rightarrow Bag(I)$, where $bag(I)$ is the multiset on the ordered set I.*
   $C(r) = C_{J(r)}$ denotes the colour domain of node r;
5. *$W^-, W^+, W^h : W^-(p,t), W^+(p,t), W^h(p,t) \in [C_{J(t)} \rightarrow Bag(C_{J(p)}]$ the input, output and inhibition functions are arc expressions;*
6. *$\Phi(t) : C_{J(t)} \rightarrow \{TRUE, FALSE\}$ is a standard predicate associated with the transition t.*
   By default we assume $\forall t \in T : \Phi(t) = TRUE$;
7. *$\pi : T \rightarrow I\!N$ the priority function.*
   By default we assume $\forall t \in T : \pi(t) = 0$;
8. *$M_0 : M_0(p) \in Bag(C(p))$ is the initial marking of p;*
9. *θ is the function defined on the set of transitions T such that $\forall t \;\; \theta : C(t) \rightarrow I\!R^+$ represents the average firing rate.*

3. AGGREGATION IN SPA: STRONG EQUIVALENCE

The aggregation technique introduced in SPA is based on an equivalence notion that can be used to partition the state space underlying a model into equivalence classes. Only the equivalence classes are considered in the aggregated state space. This equivalence notion has been called *strong equivalence* in PEPA (Hillston, 1994) and *Markovian bisimulation* in MTIPP (Hermanns & Rettelbach, 1994).

In this section we informally introduce strong equivalence, major details may be found in (Hillston, 1994);

details of Markovian bisimulation and MTIPP may be found in (Hermanns & Rettelbach, 1994). We assume that the reader is familiar with the PEPA notation.

In PEPA each component C is characterised by its *derivative set* (DS), representing the set of all the reachable states of the system, and by its *derivation graph* (DG) whose nodes are elements of the DS and whose arcs are labelled by pairs (*action type,rate*) that cause the state change. The DG provides a useful way of reasoning about the model and is the basis for the generation of the underlying continuous time Markov chain (CTMC).

The *transition rate* between two components C_i and C_j in the DG is the rate at which the system changes from behaving as component C_i to behaving as C_j. It is denoted by $q(C_i, C_j)$ and is the sum of the activity rates labelling arcs connecting node C_i to node C_j.

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

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

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

The concept of total conditional transition rate is the basis for the definition of strong equivalence since two PEPA components are considered strongly equivalent ($\cong$) if there is an equivalence relation between them such that, for any action type α, the total conditional transition rates from those components to any equivalence class, via activities of this type, are the same.

The notion of strong equivalence has been used in (Hillston, 1995) to generate a Markov process underlying a PEPA model which is aggregated. Usually the Markov process is obtained starting from the DG and, as a consequence, it may have a large state space, even for relatively simple PEPA models. However the partition formed on the DS of any PEPA model by strong equivalence may be used to provide exact aggregation (Kemeny & Snell, 1960) of the underlying CTMC. Let us consider the simple example below

$$\begin{aligned} P &\stackrel{\text{def}}{=} (\alpha, \lambda).P' \\ P' &\stackrel{\text{def}}{=} (\beta, \mu).P \\ Q &\stackrel{\text{def}}{=} P \| P \end{aligned}$$

The DS of Q is given by

$$ds(Q) = \{P\|P, (\beta,\mu).P\|P, P\|(\beta,\mu).P, (\beta,\mu).P\|(\beta,\mu).P\}$$

The two states $(\beta, \mu).P\|P$ and $P\|(\beta, \mu).P$ can perform an action of type β with the same rate μ resulting in the same derivative $P|P$ and, therefore, they cannot be distinguished by any external observer, even under repeated observation. These two states belong to the same equivalence class, which is usually denoted by an element of the class inscribed inside square brackets. Hence, the lumped DS of Q is

$$ds(Q)/_{\cong} = \{\, [P\|P],\ [P\|(\beta,\mu).P],\ [(\beta,\mu).P\|(\beta,\mu).P] \,\}$$

Starting from $ds(Q)/_{\cong}$ we can obtain the *lumped derivation graph* (LDG) of Figure 1(b) in which the two states $P\|(\beta, \mu).P$ and $(\beta, \mu).P\|P$, corresponding to strongly equivalent derivatives, have been merged into a single state.

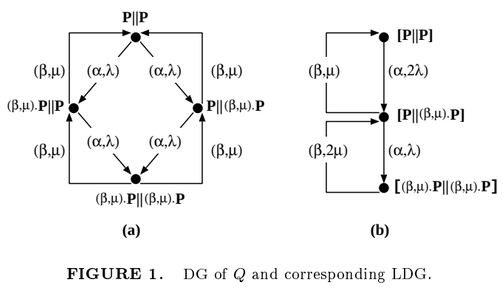


FIGURE 1. DG of Q and corresponding LDG.

A lumped model can be obtained by considering the associated LDG: the nodes represent the components of the lumped model and the labels on the arcs are the activities which the components can perform.

The computation of the aggregated CTMC is straightforward: the states of the Markov process are in one-to-one correspondence with the states of the LDG, and the transition rates between nodes are the sum of the total conditional transition rates attached to the arcs connecting them.

Unfortunately this technique, which may result in a drastic reduction of the state space of the underlying Markov process, still requires the generation of the DS of the complete model. In some cases this set can be so large that even the aggregation of the model is infeasible. However, all the benefits of this technique can be introduced by taking advantage of the fact that strong equivalence is a congruence, and by exploiting the compositional structure of the model (Hillston, 1995).

Instead of constructing the complete DS of the entire model and then partitioning it into equivalence classes, it is possible to generate *partial* DSs through considering pairs of cooperating components in turn. These partial DSs can be aggregated and the corresponding lumped component can be computed by considering the associated LDG.

Furthermore, each pair of components can be replaced by the lumped one, obtaining a new model which is strongly equivalent to the original (i.e. its behaviour is maintained) but, in general, has a smaller DS.

If we consider, for instance, the term $Q \stackrel{\text{def}}{=} (P \underset{L}{\bowtie} P) \underset{K}{\bowtie} P$, the aggregation is obtained by first computing the lumped component of $P \underset{L}{\bowtie} P$, say $\widehat{P}$, and then replacing $P \underset{L}{\bowtie} P$ by $\widehat{P}$. The lumped component of $\widehat{P} \underset{K}{\bowtie} P$ is finally computed.

Notice that the DS of the original model does not need to be constructed, and that no CTMC is derived until the aggregation procedure is complete. Details of the procedure that realises the compositional aggregation technique may be found in (Hillston, 1994).

4. SWN SEMANTICS FOR PEPA

In this section we describe by means of examples a SWN semantics for PEPA. For each language operator *op* we show the operation $op_{\mathcal{N}}$ that has to be performed at the net level to reproduce its effect. For some operators we also introduce a formal semantics, a complete formalisation will appear in a forthcoming paper.

In the following examples we assume that $\forall t \in T$: $\pi(t) = 0$ and $\Phi(t) = TRUE$ and therefore we can omit π and Φ in the formal definitions. Moreover, the basic colour classes are not partitioned into static subclasses, the colour domains cannot contain repetitions of the same basic colour class, and the successor function is not needed. Transition rates do not depend on transition instances so that the function θ is defined as $\theta : T \to \mathbb{R}^+$. Notice that the models we derive belong to a subclass of SWN known as Regular stochastic Petri nets (Dutheillet & Haddad, 1989).

Because each transition must be uniquely named and yet several transitions represent the same action name, a labelling function l is provided which maps from transitions back to action names. We thus consider labelled SWN models defined as

$$\mathcal{N} = (P, T, C, J, W^-, W^+, M_0, \theta, l)$$

For simplicity, M_0 represents the set of initial places from which we can derive the initial marking considering the fact that the initial places contain all the tokens in the corresponding colour domain. Moreover, we use the notation ${}^\bullet r$ ($r^\bullet$) to denote the nodes which are input (output) of a given node $r \in P \cup T$.

Basic element Figure 2 shows the translation of the basic PEPA element (α, r). The net structure is composed of an input place p1, a transition t1, and an output place p2. The colour class $C_i = \{c_i^k\}$ is associated with the places and the transition of the net. Usually transitions' colour domains are not explicitly represented in the graphical representation since they can be derived from the colour domains of the places connected to them.

The initial marking of p1 is equal to S, a special symbol indicating that *all* the objects in C_i (only c_i^k in this simple case) are initially in place p1. Transition t1 models the action type α and it is enabled in the initial marking. After it has fired token $\langle c_i^k \rangle$ is removed from place p1 and is added into place p2 as specified by the identity function $\langle x \rangle$ labelling input and output arcs.

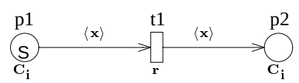


FIGURE 2. Basic element.

DEFINITION 4.1. **Basic element**
The net representation of the basic element (α, r) *is* $\mathcal{N} = (P, T, C, J, W^-, W^+, M_0, \theta, l)$ *where*

$$\begin{array}{lcl} P & = & \{p_1, p_2\} \\ T & = & \{t_1\} \\ C & = & \{C_i\} \quad and \quad C_i = \{c_i^k\} \\ J(r) & = & \{i\} \ \ \forall r \in P \cup T \\ W^- & = & W^-(p_1, t_1) = \langle x \rangle \\ W^+ & = & W^+(p_2, t_1) = \langle x \rangle \\ M_0 & = & \{p_1\} \\ \theta(t_1) & = & r \\ l(t_1) & = & \alpha \end{array}$$

Prefixing Figure 3 shows the translation of the expressions $P_1 \stackrel{\text{def}}{=} (\alpha, r_1).P_2$, $P_2 \stackrel{\text{def}}{=} (\beta, r_2).P_3$. First we translate the two components in isolation (cf. Figure 3(a)) and then we translate the "." operator (cf. Figure 3(b)) by merging the final place of the net modelling P_1 (p2), with the input place of the net of P_2 (p3). Notice that P_2 may be any language expression and hence its SWN representation may have any number of initial places. In such a case p2 has to be merged with all the initial places of the net modelling P_2.

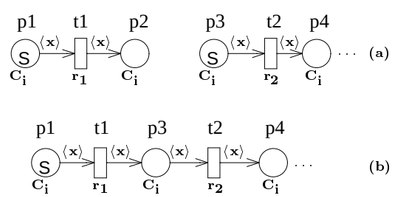
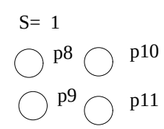


FIGURE 3. Prefixing.

DEFINITION 4.2. **Prefixing**
Let $\mathcal{N}_1 = (\{p_1, p_2\}, \{t_1\}, C_1, J_1, W_1^-, W_1^+, M_{0,1}, \theta_1, l_1)$ *be the net representing* (α, r) *and* $\mathcal{N}_2 = (P_2, T_2, C_2, J_2, W_2^-, W_2^+, M_{0,2}, \theta_2, l_2)$ *be the net representing* Q, *with* $\{p_1, p_2\} \cap P_2 = \{t_1\} \cap T_2 = \emptyset$.
The net modelling $P \stackrel{\text{def}}{=} (\alpha, r).Q$ *is* $\mathcal{N}_{(.)} =$

S= 2

S= 1



$(P, T, C, J, W^-, W^+, M_0, \theta, l)$ *where*

$$
\begin{array}{lcl}
P & = & \{p_1\} \cup P_2 \\
T & = & \{t_1\} \cup T_2 \\
C & = & C_1 \cup C_2 \\
J(r) & = & \begin{cases} J_1(r) & if\ \ r = p_1 \\ J_2(r) & if\ \ r \in P_2 \cup T_2 \\ J_1({}^\bullet r) \cup J_2(p) & if\ \ r = t_1 \wedge p \in M_{0,2} \end{cases} \\
W^-(p,t) & = & \begin{cases} W_1^-(p,t) & if\ \ p = p_1 \wedge t = t_1 \\ W_2^-(p,t) & if\ \ t \neq t_1 \end{cases} \\
W^+(p,t) & = & \begin{cases} W_2^+(p,t) & if\ \ t \neq t_1 \\ W_2^-(p,t') & if\ \ p \in M_{0,2} \wedge t = t_1 \\ & & \qquad \wedge\ t' \in p^\bullet \end{cases} \\
M_0 & = & M_{0,1} \\
\theta(t) & = & \begin{cases} \theta_1(t) & if\ \ t = t_1 \\ \theta_2(t) & if\ \ t \in T_2 \end{cases} \\
l(t) & = & \begin{cases} l_1(t) & if\ \ t = t_1 \\ l_2(t) & if\ \ t \in T_2 \end{cases}
\end{array}
$$

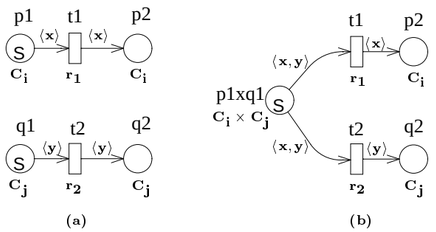


FIGURE 4. Choice.

Choice Let us consider the term $P_1 + Q_1$ where $P_1 \stackrel{\text{def}}{=} (\alpha, r_1).P_2$, $Q_1 \stackrel{\text{def}}{=} (\beta, r_2).Q_2$. The choice operator is translated starting from the individual subnets representing the two components. The subnet in the upper part of Figure 4(a) represents P_1. The colour domain of the places and the transition is $C_i = \{c_i^k\}$ and the identity function $\langle x \rangle$ labels the arcs. The subnet in the lower part represents Q_1. In this case the colour domain is $C_j = \{c_j^l\}$ and the function on the arcs is $\langle y \rangle$.

The net of $P_1 + Q_1$ in Figure 4(b) is obtained by considering the Cartesian product of the input places `p1` and `q1`. The colour domain of place `p1xq1` has to be changed to reflect the Cartesian product operation at the colour structure level. This is simply obtained by associating with place `p1xq1` a colour domain equal to the Cartesian product of C_i and C_j. Place `p1xq1` now contains structured tokens consisting of pairs of basic objects $\langle c_i^k, c_j^l \rangle$ with $c_i^k \in C_i$ and $c_j^l \in C_j$. The initial marking S denotes all the pairs belonging to $C_i \times C_j$ (in this case only $\langle c_i^k, c_j^l \rangle$).

The colour domains of transitions `t1` and `t2` must also be changed to be consistent with the colour domains of the corresponding input place. This fact is reflected in the modified function $\langle x, y \rangle$ labelling their input arcs.

Transitions `t1` and `t2` are in conflict in the initial marking and the first that fires determines the state change. If, for instance, transition `t1` fires first, token $\langle c_i^k, c_j^l \rangle$ is removed from `p1xq1` and token $\langle c_i^k \rangle$ is added into `p2`. This is obtained by labelling the arc connecting transition `t1` to place `p2` with the function $\langle x \rangle$ meaning that only the first component of the pair $\langle c_i^k, c_j^l \rangle$ has to be added in `p2`. The same reasoning holds for `t2`, but considering the second element of $\langle c_i^k, c_j^l \rangle$.

DEFINITION 4.3. **Choice**
Let $\mathcal{N}_1 = (P_1, T_1, C_1, J_1, W_1^-, W_1^+, M_{0,1}, \theta_1, l_1)$ *and* $\mathcal{N}_2 = (P_2, T_2, C_2, J_2, W_2^-, W_2^+, M_{0,2}, \theta_2, l_2)$ *with* $P_1 \cap P_2 = T_1 \cap T_2 = C_1 \cap C_2 = \emptyset$ *be the nets modelling two components* P *and* Q. *We define the net modelling the component* $R \stackrel{\text{def}}{=} P + Q$ *as* $\mathcal{N} = (P, T, C, J, W^-, W^+, M_0, l, \theta)$ *where*

$$
\begin{array}{lcl}
P & = & P_1 \cup P_2 - \{M_{0,1} \cup M_{0,2}\} \cup \{M_{0,1} \times M_{0,2}\} \\
T & = & T_1 \cup T_2 \\
C & = & C_1 \cup C_2 \\
J(r) & = & \begin{cases} J_1(p) \cup J_2(q) & if\ r = p \times q \in M_{0,1} \times M_{0,2} \\ J_1(r) & if\ r \in P_1 - M_{0,1} \\ J_2(r) & if\ r \in P_2 - M_{0,2} \\ J_1(r) & if\ r \in T_1 - M_{0,1}^\bullet \\ J_2(r) & if\ r \in T_2 - M_{0,2}^\bullet \\ J_1(p) \cup J_2(q) & if\ p \times q \in M_{0,1} \times M_{0,2} \wedge \\ & \qquad ((r \in T_1 \cap p^\bullet) \vee (r \in T_2 \cap q^\bullet)) \end{cases} \\
W^-(p,t) & = & \begin{cases} W_1^-(p,t) & \forall p \in P_1 - M_{0,1}, \forall t \in T_1 - M_{0,1}^\bullet \\ W_2^-(p,t) & \forall p \in P_2 - M_{0,2}, \forall t \in T_2 - M_{0,2}^\bullet \\ \langle W_1^-(q_1, t_1), W_2^-(q_2, t_2) \rangle & \\ \qquad if\ \ p = q_1 \times q_2 \in M_{0,1} \times M_{0,2} \wedge & \\ \qquad (t = t_1 \vee t = t_2) \wedge (t_1 \in q_1^\bullet \wedge t_2 \in q_2^\bullet) & \end{cases} \\
W^+(p,t) & = & \begin{cases} W_1^+(p,t) & \forall p \in P_1 - M_{0,1}, \forall t \in T_1 \\ W_2^+(p,t) & \forall p \in P_2 - M_{0,2}, \forall t \in T_2 \end{cases} \\
M_0 & = & M_{0,1} \times M_{0,2} \\
\theta(t) & = & \begin{cases} \theta_1(t) & if\ \ t \in T_1 \\ \theta_2(t) & if\ \ t \in T_2 \end{cases} \\
l(t) & = & \begin{cases} l_1(t) & if\ \ t \in T_1 \\ l_2(t) & if\ \ t \in T_2 \end{cases}
\end{array}
$$

Definition 4.3 does not cover the case of a choice between two replicas of the same component, $P \stackrel{\text{def}}{=} P_1 + P_1$, since we have the condition $C_1 \cap C_2 = \emptyset$. We can use Definition 4.3 for this case if we delete the condition $C_1 \cap C_2 = \emptyset$ and allow colour domains with repetitions of the same colour class. Another solution, which leads to simpler models, is that of merging the input places of the nets modelling the replicas of P_1 without considering the Cartesian product of their colour domains.

Cooperation The translation of $P_1 \| Q_1$ where $P_1 \stackrel{\text{def}}{=} (\alpha, r_1).P_2$ and $Q_1 \stackrel{\text{def}}{=} (\alpha, r_2).Q_2$ is shown in Figure 5(a). As before, P_1 and Q_1 are first translated in isolation. The colour domains are $C_i = \{c_i^k\}$ for P_1 and $C_j = \{c_j^l\}$ for Q_1, and we use different variables for the identity functions in the two components. The net of the whole system is given by the union of the two subnets.

Figure 5(b) shows the translation of the cooperation between two components when they execute actions of

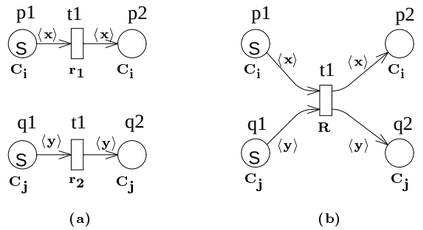


FIGURE 5. Parallel composition and cooperation.

the same type belonging to the cooperation set, i.e. $(\alpha, r_1).P_2 \underset{\{\alpha\}}{\bowtie} (\alpha, r_2).Q_2$. The cooperation is obtained by merging the transitions modelling the actions α into a single one. The colour domain of t1 is now the Cartesian product of C_i and C_j because it is determined from the colour domains of its input places.

The rate R to be associated with transition t1 is computed using the formula (Hillston, 1994)

$$R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} min(r_\alpha(P), r_\alpha(Q))$$

$r_\alpha(P)$ is called the *apparent rate* of action type α in the P component and it is the rate at which α appears to an external observer. It is assumed that when two components carry out α in cooperation, their total capacity to complete the activity is limited to the capacity of the slowest, i.e. $min(r_\alpha(P), r_\alpha(Q))$. Moreover, since in general each component may enable several α activities, it is also assumed that each independently chooses which instance takes part in the cooperation. To reflect these choices the rate R is multiplied by the product of the probabilities that each contributing activity is selected, i.e. $\frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)}$.

In this simple case each component enables a single α activity and therefore the rate of t1 is

$$R = \frac{r_1}{r_1} \frac{r_2}{r_2} min(r_1, r_2) = min(r_1, r_2)$$

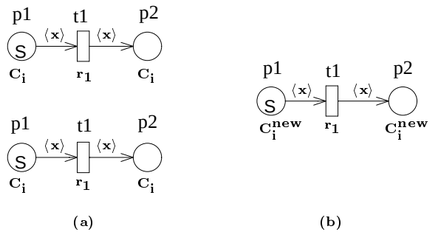


FIGURE 6. Parallel composition of two equal components.

Let us now consider other examples of parallel composition and cooperation to see how they can be mapped into more compact SWN models. The expression $P_1 \| P_1$ where $P_1 \stackrel{\text{def}}{=} (\alpha, r_1).P_2$ could be translated by considering two replicas of the same net separately (cf. Figure 6(a)), but there is also another translation possibility in which the two identical nets are folded into a single one as shown in Figure 6(b). In order to distinguish them we need to redefine the basic colour class C_i as $C_i^{new} = \{c_i^{k_1}, c_i^{k_2}\}$. The initial marking of place p1, equal to S, denotes all the objects in C_i^{new}, so that place p1 now contains two tokens, $\langle c_i^k \rangle$ and $\langle c_i^2 \rangle$.

As a general rule, we can say that when we want to add new replicas of the same component we can either modify the net structure by replicating identical subnets without changing the colour structure, or we can keep the same net structure and modify the colour structure so that the new replica is taken into account at the level of the colour class definition. We prefer the second solution because it leads to more compact SWN models whose symmetries can be automatically exploited during the state space generation.

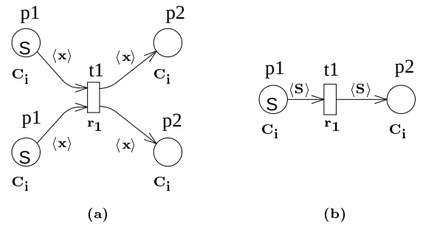


FIGURE 7. Cooperation between two equal components.

For the expression $P_1 \underset{\{\alpha\}}{\bowtie} P_1$ where $P_1 \stackrel{\text{def}}{=} (\alpha, r_1).P_2$, we also have two possibilities. In Figure 7(a) the cooperation is modelled by considering the two replicas of P_1 separately. In Figure 7(b) a more compact net structure is shown. The colour class C_i is again defined as $C_i^{new} = \{c_i^{k_1}, c_i^{k_2}\}$ to distinguish the two replicas, and the synchronisation function $\langle S \rangle$ labels the input and output arcs of transition t1. The transition is enabled if *all* the tokens in class C_i^{new} are present in place p1, modelling a situation in which both the replicas are ready to cooperate.

As before, we can formulate a general rule for modelling the cooperation between identical components. We can either replicate the net structure and merge the transitions modelling any shared action or we can modify the colour class definition adding new elements corresponding to the new components. In the second case we must also label arcs with the synchronisation function $\langle S \rangle$ to obtain the correct behaviour. Again we prefer the second solution for the same reason as before.

Hiding For the term P/L we have two different translations, depending whether action type α belongs to the

set L or not. The net representation of P/L is obtained by starting from the net of P and renaming with τ all the transitions with a label $\alpha \in L$. Notice that different τ transitions have to be distinguished since they model internal delays.

Recursion PEPA does not provide an explicit recursion operator and infinite behaviours can be modelled using constants. Figure 8 shows the translation of the term $P \stackrel{\text{def}}{=} (\alpha, r_1).(\beta, r_2).P$, which is performed in two steps. First we translate the body of the recursion (Figure 8(a)) and then we close the net by folding places `p1` and `p3` to model an infinite behaviour (Figure 8(b)). The colour domain of places and transitions is $C_i = \{c_i^k\}$ and the arcs are labelled with the identity function $\langle x \rangle$.

The translation of recursive term is much more complex in general. We consider only this simple schema because it is the one that we will use in the models described in the next sections. A more formal treatment of recursion has to be addressed.

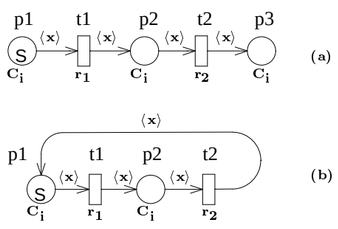


FIGURE 8. Recursion.

Discussion The SWN models derived by applying the previous semantic rules may have a number of colour classes and a number of function variables which increase with the number of components of the PEPA model we are translating. However, these models can be simplified taking advantage of decolourisation techniques (Chiola & Franceschinis, 1991) when the colour classes and the colour functions are proved to be unnecessary.

We need to investigate the SWN semantics to understand when colour domains and colour functions are really needed. For instance, it seems that colours are necessary when translating the cooperation between components (Figure 5,6,7), while they can be avoided during the translation of the basic element (Figure 2) and the prefixing operator (Figure 3). Since the SWN formalism allows the use of neutral places and neutral transitions, we can think of new rules that generate nets which have both neutral and coloured nodes.

If the semantics is well defined, the SWN model of a term P obtained using the semantics given here plus the decolourisation techniques, and the SWN model obtained using the new rules should be the same, i.e. nets without any redundant colour annotation.

5. RELATIONSHIP BETWEEN THE LDG AND THE SRG

In the remaining part of this paper we discuss the relationship that exists between the LDG underlying a PEPA model and the SRG underlying the corresponding SWN model.

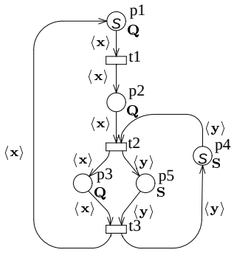
Starting from a PEPA model, say P, it is possible to compute the corresponding aggregated PEPA model $\hat{P}$ and the underlying LDG. Applying the semantic rules given in Section 4., it is also possible to derive the SWN model, $\mathcal{N}[\![P]\!]$, corresponding to P and to compute its SRG. In Section 5.1. we apply this procedure to show that the LDG of a random polling system modelled with PEPA and the SRG of the SWN model derived by applying the SWN semantics are isomorphic.

Our initial goal was to show the existence of an isomorphism relation between the LDG of any PEPA model and the SRG of the corresponding SWN model in order to conclude that the two techniques achieve the same aggregation on the underlying state space. Unfortunately this is not true in the general case as we discuss in Section 5.2. describing a small example for which the aggregated state spaces are not isomorphic. We cannot therefore conclude that the two techniques lead to an isomorphism between LDG and the SRG, but we can obtain a weaker result showing that the LDG and the SRG are instead bisimilar.

5.1. Random polling system example

The system we model is a random polling system with three queues and two servers and, for simplicity, let us assume that each queue has capacity one. The PEPA specification and the SWN model are shown in Figure 9.

$$
\begin{aligned}
Q_0 &\stackrel{\text{def}}{=} (arrive, \lambda).Q_1 \\
Q_1 &\stackrel{\text{def}}{=} (walk, \top).Q_2 \\
Q_2 &\stackrel{\text{def}}{=} (service, \top).Q_0 \\
\\
S_0 &\stackrel{\text{def}}{=} (walk, w).S_1 \\
S_1 &\stackrel{\text{def}}{=} (service, \mu).S_0
\end{aligned}
$$



$$Sys \stackrel{\text{def}}{=} (Q_0 \| Q_0 \| Q_0) \underset{\{service, walk\}}{\bowtie} (S_0 \| S_0)$$

FIGURE 9. PEPA and SWN models of the random polling system.

Each customer that arrives at a queue (action *arrive*)

starts waiting for a server. When a server is polling the queue in which a customer is waiting a shared action *walk* is performed. Then the service can be provided thanks to the shared action *service*. Notice that actions *walk* and *service* are passive in the queue components and hence their rates are determined by the rates of the corresponding active actions performed by the server components.

In the corresponding SWN model we have two colour classes $Q = \{q_1, q_2, q_3\}$ and $S = \{s_1, s_2\}$ modelling the queues and the servers respectively. The transition labels are $l(\mathsf{t1}) = arrive$, $l(\mathsf{t2}) = walk$, $l(\mathsf{t3}) = service$ and the transition rates are $\theta(\mathsf{t1}) = \lambda$, $\theta(\mathsf{t2}) = w$, $\theta(\mathsf{t3}) = \mu$.

The LDG underlying the PEPA specification, shown in Figure 10, has been obtained by finding strongly equivalent states as discussed in Section 3.. The SRG of the SWN model, which has been automatically obtained, contains nine symbolic markings and it is isomorphic to the LDG of Figure 10. That is, it has the same number of states (each symbolic marking in the SRG corresponds to a node in the LDG), as well as the same number of transitions. The underlying Markov processes are also isomorphic.

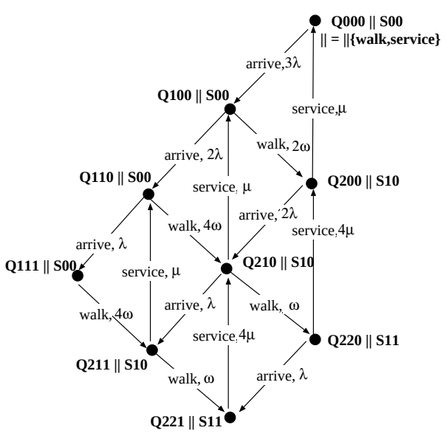


FIGURE 10. Aggregated DG of *Sys*.

5.2. The strong equivalence "lumps more" with respect to the SRG

Finding an isomorphism relation between the LDG underlying *any* PEPA model and the SRG underlying the corresponding SWN model would imply that both formalisms achieve the same aggregation on the underlying state space. Unfortunately such an isomorphism relation cannot be established in the general case as shown by the small example below.

$$\begin{aligned} P_1 &\stackrel{\text{def}}{=} (\alpha, 2\lambda).P_2 \\ P_2 &\stackrel{\text{def}}{=} (\beta, \mu).P_3 \\ P_3 &\stackrel{\text{def}}{=} (\alpha, \lambda).P_2 + (\alpha, \lambda).P_4 \\ P_4 &\stackrel{\text{def}}{=} (\beta, \mu).P_1 \end{aligned}$$

The derivative set of P_1 is

$$ds(P_1) = \{\ (\alpha, 2\lambda).P_2,\ (\beta, \mu).P_3,\ (\alpha, \lambda).P_2 + (\alpha, \lambda).P_4, (\beta, \mu).P_1\ \}$$

and the corresponding DG is shown in Figure 11(a).

The two derivatives $(\alpha, 2\lambda).P_2$ and $(\alpha, \lambda).P_2 + (\alpha, \lambda).P_4$ are strongly equivalent since they perform actions of the same type α, with the same conditional transition rate (equal to 2λ), and the states reached after the completion of these activities are in the same equivalence class. Also $(\beta, \mu).P_3$ and $(\beta, \mu).P_1$ are strongly equivalent for the same reason. We can thus partition $ds(P_1)$ into two equivalence classes

$$\begin{aligned} [P_1] &= \{\ \underbrace{(\alpha, 2\lambda).P_2}_{P_1}, \underbrace{(\alpha, \lambda).P_2 + (\alpha, \lambda).P_4}_{P_3}\ \} \\ [P_2] &= \{\ \underbrace{(\beta, \mu).P_3}_{P_2}, \underbrace{(\beta, \mu).P_1}_{P_4}\ \} \end{aligned}$$

from which we can derive the LDG shown in Figure 11(b). Each state in the graph corresponds to a pair of strongly equivalent states, that is $P_1 \cong P_3$ and $P_2 \cong P_4$.

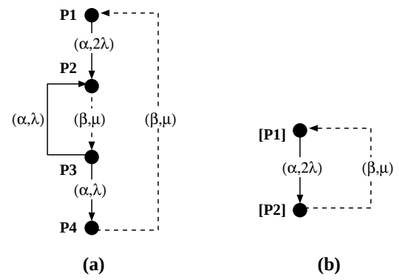
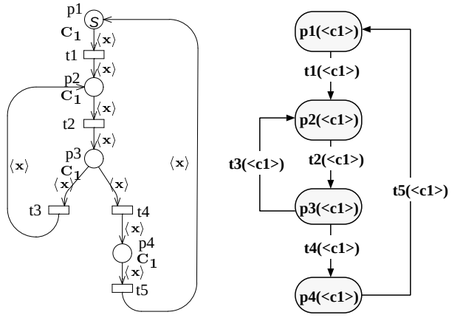


FIGURE 11. DG and LDG of P_1.

The SWN model corresponding to P_1 is shown in Figure 12(a). The colour domain associated with places and transitions is $C_1 = \{c_1\}$ and the initial marking of place p1 is equal to $\langle c_1 \rangle$. The mapping between transitions and action names is $l(\mathsf{t1}) = l(\mathsf{t3}) = l(\mathsf{t4}) = \alpha$, $l(\mathsf{t2}) = l(\mathsf{t5}) = \beta$; the transition rates are $\theta(\mathsf{t1}) = 2\lambda$, $\theta(\mathsf{t2}) = \theta(\mathsf{t5}) = \mu$, $\theta(\mathsf{t3}) = \theta(\mathsf{t4}) = \lambda$.

The underlying SRG, shown in Figure 12(b), has four symbolic markings (represented considering the non-empty places only) and it is isomorphic to the DG of P_1. However it is not possible to obtain a more aggregated state space isomorphic to the LDG of Figure 11(b). For instance, the two markings $P_1(\langle c_1 \rangle)$ and $P_3(\langle c_1 \rangle)$ which

correspond to strongly equivalent states in the PEPA model, do not belong to the same symbolic marking in the SWN model. Marking $P_1(\langle c_1 \rangle)$ enables transition t1, marking $P_3(\langle c_1 \rangle)$ enables transitions t3 and t4, and $l(\mathsf{t1}) = l(\mathsf{t3}) = l(\mathsf{t4}) = \alpha$. However this is not enough for the two markings to belong to the same symbolic marking: they enable transitions which are different although their associated labels are the same.



(a)

(b)

FIGURE 12. SWN model and SRG of P_1.

This is a consequence of the difference between actions and transitions. PEPA components can perform the same action several times and the states that enable actions of the same type can be lumped together, under the condition that these actions are executed with the same rates and that the resulting states are also equivalent.

In the SWN formalism (and in Petri nets in general) transitions must have unique names and only those markings that enable exactly the same transition, but for different colour instances, can be lumped together to form a symbolic marking.

5.3. Horizontal and vertical aggregation

Let us now consider the model P consisting of two replicas of P_1 whose DG is shown in Figure 13. The states $P_1 \| P_2$ and $P_2 \| P_1$ are strongly equivalent, and therefore belong to the same equivalence class which we denote as $[P_1 \| P_2]$. The same reasoning holds for all the pairs of states like $P_1 \| P_3$ and $P_3 \| P_1$, $P_1 \| P_4$ and $P_4 \| P_1$, etc. in which the components appear in a different order.

If we consider only one element (the equivalence class) for each pair of equivalent states, we obtain the DG depicted in Figure 14. This kind of reduction takes into account only one among different, but equivalent, interleavings. For instance, we consider only one path from the initial state to $P_2 \| P_2$, i.e. only one between the two paths $P_1 \| P_1 \to P_1 \| P_2 \to P_2 \| P_2$ and $P_1 \| P_1 \to P_2 \| P_1 \to P_2 \| P_2$.

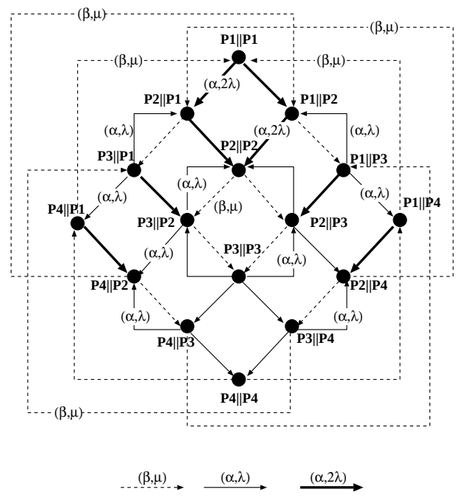


FIGURE 13. DG of $P = P_1 \| P_1$.

Note that, since in the DG of Figure 13 there are two arcs labelled $(\alpha, 2\lambda)$ exiting from the initial state (and entering into equivalent states) the total conditional transition rate for action type α is 4λ. This is reflected in the DG by the rate 4λ associated with action type α in the arc from $P_1 \| P_1$ to $[P_1 \| P_2]$.

Graphically this kind of aggregation that takes into account only one among equivalent interleavings can be seen as a *horizontal* aggregation in which equivalent nodes at the same level are folded together. It is easy to see such folding on the DG comparing Figure 13 and Figure 14.

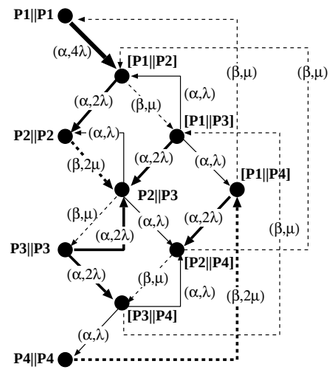


FIGURE 14. DG of P after horizontal aggregation.

The strong equivalence relation also allows another

kind of reduction of the state space which takes into account repeated patterns of behaviour. Let us call this reduction *vertical* aggregation as opposed to the horizontal one. In this case we consider as belonging to the same equivalence class those states characterised by the same pattern of behaviour; that is, states that after performing the same actions, with the same rates, reach equivalent states. We need to find states characterised by arcs labelled with the same action types and such that the total transition rates for each type are the same. These states belong to the same equivalence class meaning that the state space can be further aggregated.

We know that $P_1 \cong P_3$ and $P_2 \cong P_4$ (see the example of P_1 and Figure 11) and, using the fact that the strong equivalence relation is a congruence, the following equivalences also hold.

$$P_1 \| P_1 \cong P_1 \| P_3 \; (P_3 \| P_1) \cong P_3 \| P_3$$
$$P_1 \| P_2 \; (P_2 \| P_1) \cong P_1 \| P_4 \; (P_4 \| P_1) \cong P_2 \| P_3 \; (P_3 \| P_2) \cong P_3 \| P_4 \; (P_4 \| P_3)$$
$$P_2 \| P_2 \cong P_2 \| P_4 \; (P_4 \| P_2) \cong P_4 \| P_4$$

On the basis of these equivalences we find three equivalence classes

1) $[P^{'}] = \{ P_1 \| P_1, [P_1 \| P_3], P_3 \| P_3 \}$
2) $[P^{''}] = \{ [P_1 \| P_2], [P_1 \| P_4], [P_2 \| P_3], [P_3 \| P_4] \}$
3) $[P^{'''}] = \{ P_2 \| P_2, P_4 \| P_4, [P_2 \| P_4] \}$

The LDG associated with $P \stackrel{\text{def}}{=} P_1 \| P_1$ is finally shown in Figure 15(b). It contains only three states, one for each equivalence class. To better understand this vertical aggregation we have included in the left hand side of Figure 15 the DG of Figure 14 in which nodes are now labelled with the equivalence class to which they belong. Comparing these two graphs it is possible to recognise the repeated behaviour which is considered only once in the LDG in the right hand side of the figure.

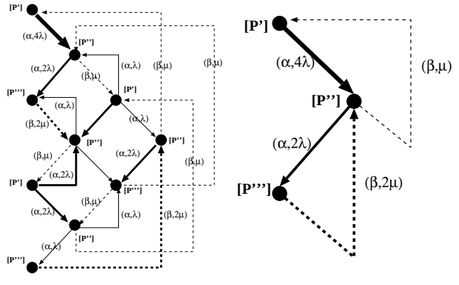


FIGURE 15. LDG of P after horizontal and vertical aggregation.

Notice that the use of the strong equivalence relation to partition the state space of a PEPA model results in both horizontal and vertical aggregations being performed at the same time. The distinction between these two different types of aggregation is however very helpful in understanding the different results achieved by PEPA and SWN.

The SWN model of $P_1 \| P_1$ is obtained by changing the colour class C_1 into $C_1^{new} = \{c_1, c_2\}$ to model the two instances of the same component. The SRG of the modified net contains ten symbolic markings and it is isomorphic to the derivation graph of Figure 14, but no further reduction can be obtained on it.

In summary, in the horizontal aggregation only one out of all the possible, but equivalent, interleavings is considered while in the vertical aggregation repeated patterns of behaviour are recognised and the states that have the same behaviour in each pattern are lumped together. The PEPA formalism allows both kinds of aggregations while the SWN formalism does not. In fact, only markings that enable the same transitions for different colour instances are lumped together, and this corresponds to the horizontal aggregation which is the only possible in SWNs. This explains the title of the section, when we said that strong equivalence *lumps more.*

We cannot find an isomorphism between LDGs and SRGs and only a strong bisimulation relation, derived from the strong bisimulation relation between labelled transition systems (Milner, 1989), can be established.

In untimed process algebra two agents are strongly bisimilar if any α action of one can be matched by an α action of the other; moreover the process terms obtained after executing the α action are still strongly bisimilar. In SPA the notion of strong bisimulation has been extended (Hillston, 1994; Hermanns & Rettelbach, 1994) to include transition rates as well as action names. In the definition below, taken form (Hillston, 1994), $\mathcal{C}$ denotes the set of components and $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$ denotes the set of activities defined as pairs (α, r), where $\alpha \in \mathcal{A}$ is the action name (type) and $r \in \mathbb{R}^+$ is the action rate.

DEFINITION 5.1. *A binary relation, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, over components is a strong bisimulation if $(P, Q) \in \mathcal{R}$ implies, for all $a = (\alpha, r) \in \mathcal{A}ct$,*

1. *$r_\alpha(P) = r_\alpha(Q)$;*
2. *whenever $P \xrightarrow{a} P'$, then for some Q', $Q \xrightarrow{a} Q'$, and $(P', Q') \in \mathcal{R}$;*
3. *whenever $Q \xrightarrow{a} Q'$, then for some P', $P \xrightarrow{a} P'$, and $(P', Q') \in \mathcal{R}$.*

Starting from the notion of strong bisimulation between labelled transition system, and observing that the LDG and the SRG are labelled transition systems, we can state the following proposition.

PROPOSITION 5.1. *The LDG underlying a PEPA model P and the SRG underlying the corresponding SWN model are strongly bisimilar.*

5.4. Canonical form for PEPA terms

In the previous example we have identified two types of aggregation, that we have called horizontal and vertical.

In this section we focus on the horizontal aggregation giving some preliminary ideas that can be applied to formally define an algorithm for finding it.

We need criteria to decide

1. when the derivatives are strongly equivalent;
2. how to express the representative interleaving.

The first problem is solved by considering the equational laws developed for PEPA (Hillston, 1994). For instance, we have implicitly used the commutative law $P \underset{L}{\bowtie} Q \cong Q \underset{L}{\bowtie} P$, to establish that $P_1 \| P_2 \cong P_2 \| P_1$.

The second problem is solved by introducing the concept of *ordered* representation. The ordering criterion has to be defined so that all the strongly equivalent interleavings have the same ordered representation. Using the lexicographic ordering we can find such a unique representation by looking for the representation which is *lexicographically minimal*. For example, if $P_1 < P_2$ we have that $P_1 \| P_2 < P_2 \| P_1$.

Consider the PEPA system below, consisting of several replicas of cooperating components

$$Sys \stackrel{\text{def}}{=} (P_i \underset{L_i}{\bowtie} \ldots P_i) \underset{K_1}{\bowtie} (P_j \underset{L_j}{\bowtie} \ldots P_j) \underset{K_2}{\bowtie} (P_k \underset{L_k}{\bowtie} \ldots P_k)$$

We call symmetric components those subcomponents composed of syntactically indistinguishable replicas, in this case those inside round brackets. Two derivatives are strongly equivalent if they can be obtained by permutation inside subcomponents containing identical replicas. For example, the following expressions

$$(P_i^{'} \underset{L_i}{\bowtie} \ldots P_i) \underset{K_1}{\bowtie} (P_j^{'} \underset{L_j}{\bowtie} \ldots P_j) \underset{K_2}{\bowtie} (P_k^{'} \underset{L_k}{\bowtie} \ldots P_k)$$

$$(P_i \underset{L_i}{\bowtie} \ldots P_i^{'}) \underset{K_1}{\bowtie} (P_j \underset{L_j}{\bowtie} \ldots P_j^{'}) \underset{K_2}{\bowtie} (P_k \underset{L_k}{\bowtie} \ldots P_k^{'})$$

are equivalent derivatives. Moreover, if we assume that

$$P_i < P_i^{'}, \quad P_j < P_j^{'} \quad \text{and} \quad P_k < P_k^{'}$$

the second derivative is minimal.

The ideas behind the minimal representation of a derivative are analogous to those introduced in the SWN formalism for the computation of a *canonical* representation for the symbolic marking. Hence, using the SWN terminology, we say that the minimal derivative is in *canonical* form.

Having in mind this notion of minimal representation we can define some of the steps necessary to form an horizontally aggregated derivative set. Let us suppose to have a PEPA model P composed of symmetric subcomponents. The steps to be performed to form the aggregated derivative set $\tilde{ds}(P)$ are the following

1. for each derivative P_i find the canonical form $\tilde{P}_i$ by permuting components inside symmetric subcomponents until the ordered representation is obtained;
2. compare $\tilde{P}_i$ with all the derivatives already included in $\tilde{ds}(P)$;
3. add $\tilde{P}_i$ in to $\tilde{ds}(P)$ if it is not present.

These steps have to be repeated until no new elements can be added into the aggregated derivative set. At the end $\tilde{ds}(P)$ contains only canonical interleavings.

Notice that the steps above can only be applied to a special class of models consisting of symmetric components while, in general, we want to consider more complex language expressions. Moreover, different components may behave in the same way although their specifications are different. The terms $P \stackrel{\text{def}}{=} (\alpha, \lambda).(\beta, \mu).P$ and $Q \stackrel{\text{def}}{=} (\alpha, \lambda).(\beta, \mu).Q$ are strongly equivalent and the DG of $P \| Q$ has pairs of lumpable states that can be detected by applying the strong equivalence relation. On the other side, the algorithm for the computation of the canonical form of PEPA terms will fail in finding the corresponding equivalent interleavings since P and Q are not identical replicas. All these aspects must be considered in order to have a satisfactory version of the algorithm.

The novelty of the algorithm is that it avoids storing all the derivatives: instead of computing the whole state space and then aggregating it on the basis of the strong equivalence relation, we obtain a derivative set which is already aggregated.

Unfortunately the result is less general than the current procedure for PEPA since vertical aggregation is not achieved. Moreover it requires the computation of a canonical form each time a derivative is encountered in order to compare it with those already included in the derivative set. The computation of such canonical forms is expensive because it is based on the permutation of identical components. However, since the state space of the model may be drastically reduced, we think that this approach still remains advantageous, specially in the case of large models.

6. CONCLUSION AND FUTURE WORK

SWNs automatically detect the intrinsic symmetries of a model by means of the construction of the SRG. The SRG algorithm fails in producing a smaller description than the usual RG if the system to be modelled does not contain any symmetry, or if the model does not take into account properly the intrinsic symmetries of the system.

The algorithm for the canonical form of PEPA terms allows the detection of the same symmetries that are captured by the SRG. If we apply this algorithm to a symmetric PEPA model we obtain an underlying LDG which is isomorphic to the SRG underlying the corresponding SWN model.

However, we have shown in Section 5.2. that strong equivalence lumps more, since it can find aggregations that cannot be detected by the SRG construction algorithm. Strong equivalence offers more flexibility in this sense because it allows to detect symmetries which are hidden in the behaviour of the model, being determined by particular combinations of rates. An interesting area

for future work is the identification of the cases in which each of the procedures excels.

Further work is necessary in different directions. Firstly, the formal SWN semantics has to be defined for all the language operators, including the recursion. The definition of new rules that lead to reduced SWN models has also to be addressed.

Secondly, most of the details of the algorithm for the computation of the canonical form of PEPA (SPA) terms should be completed before thinking to an implementation in the PEPA Workbench tool (Gilmore & Hillston, 1994). Formulas to compute the number of ordinary interleavings corresponding to each canonical interleaving have to be defined too, so that the size of the ordinary state space can be obtained without generating it.

Acknowledgements The author would like to thank Jane Hillston and Stephen Gilmore for the helpful discussion.

This work is supported in part by the ESPRIT-BRA project No. 7269 "QMIPS", by a grant from The British Council and MURST-CRUI and by the CNR coordinate project PROCRIT.

References

Bernardo, M., Donatiello, L., & Gorrieri, R. 1994. Modelling and Analyzing Concurrent Systems with MPA. *In:* Herzog, U., & Rettelbach, M. (eds), *Proc.* 2^{nd} *Workshop on Process Algebra and Performance Modelling.*

Buchholz, P. 1994. Markovian Process Algebra: Composition and Equivalence. *In:* Herzog, U., & Rettelbach, M. (eds), *Proc.* 2^{nd} *Workshop on Process Algebra and Performance Modelling.*

Chiola, G., & Franceschinis, G. 1991. A structural colour simplification in Well-Formed coloured nets. *Pages 144–153 of: Proc.* 4^{th} *Intern. Workshop on Petri Nets and Performance Models.* Melbourne, Australia: IEEE-CS Press.

Chiola, G., Dutheillet, C., Franceschinis, G., & Haddad, S. 1990 (June). On Well-Formed Coloured Nets and their Symbolic Reachability Graph. *In: Proc.* 11^{th} *Intern. Conference on Application and Theory of Petri Nets.* Reprinted in *High-Level Petri Nets. Theory and Application*, K. Jensen and G. Rozenberg (editors), Springer Verlag, 1991.

Chiola, G., Dutheillet, C., Franceschinis, G., & Haddad, S. 1991. Stochastic Well-Formed Coloured Nets and Multiprocessor Modelling Applications. *In:* Jensen, K., & Rozenberg, G. (eds), *High-Level Petri Nets. Theory and Application.* Springer Verlag.

Chiola, G, Gaeta, R., & Ribaudo, M. 1992. Designing an efficient tool for Stochastic Well-Formed Coloured Petri Nets. *Pages 308–311 of:* Pooley, R., & Hillston, J. (eds), *Computer Performance Evaluation: Techniques and Tools.* Edinburgh, UK: Edinburgh University Press.

Chiola, G., Dutheillet, C., Franceschinis, G., & Haddad, S. 1993. Stochastic Well-Formed Coloured Nets for Symmetric Modelling Applications. *IEEE Transactions on Computers*, **42**(11).

Chiola, G., Franceschinis, G., Gaeta, R., & Ribaudo, M. 1995. *GreatSPN1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets.* To appear in Performance Evaluation: special issue on Performance Modelling Tools.

Dutheillet, C., & Haddad, S. 1989 (June). Regular Stochastic Petri Nets. *In: Proc. 10th Intern. Conf. Application and Theory of Petri Nets.* Reprinted in *High-Level Petri Nets. Theory and Application*, K. Jensen and G. Rozenberg (editors), Springer Verlag, 1991.

Gilmore, S., & Hillston, J. 1994. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. *In: Proc. Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation.*

Gotz, N., Herzog, U., & Rettelbach, M. 1994. Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebra. *In: Tutorial Proc. Performance 1993.* LNCS, vol. 729.

Hermanns, H., & Rettelbach, M. 1994. Syntax, Semantics, Equivalences, and Axioms for MTIPP. *In:* Herzog, U., & Rettelbach, M. (eds), *Proc.* 2^{nd} *Workshop on Process Algebra and Performance Modelling.*

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

Hillston, J. 1995 (January). Compositional Markovian Modelling Using A Process Algebra. *In: Proc. 2nd International Workshop on the Numerical Solution of Markov Chains.*

Kemeny, J.G., & Snell, J.L. 1960. *Finite Markov Chains.* Princeton, NJ: Van Nostrand.

Milner, R. 1989. *Communication and Concurrency.* Prentice Hall.

Molloy, M.K. 1982. Performance Analysis using Stochastic Petri Nets. *IEEE Transaction on Computers*, **31**(9), 913–917.