EXPLOITING SYMMETRIES IN STOCHASTIC PROCESS ALGEBRAS

Holger Hermanns
Universität Erlangen-Nürnberg
IMMD 7
hrherman@informatik.uni-erlangen.de

Marina Ribaudo
Universitá degli Studi di Torino
Dipartimento di Informatica
marina@di.unito.it

KEYWORDS

Process Algebra, Markov Chains, Compositionality, Model Reduction, Bisimulation

ABSTRACT

Stochastic Process Algebras have been introduced to enable compositional performance analysis of parallel and distributed systems. As with other high level modelling formalisms, state space explosion is a frequently observed problem, especially if the system consists of many cooperating components. However, if the components are identical replicas of each others, the state space can be reduced by means of equivalence preserving aggregation. This paper introduces symmetric parallel composition, an operator to specify sets of identical replicas cooperating in parallel. Its operational semantics is consistent with usual parallel composition whereas the state space explosion problem is drastically reduced. We illustrate this beneficial effect, and provide an interpretation of symmetric parallel composition in terms of Petri Nets.

INTRODUCTION

Continuous time Markov chains (CTMC), which are widely used as performance models in many diverse areas, are usually generated from high level descriptions such as Generalized Stochastic Petri Nets (GSPN) (Ajmone Marsan *et al.*, 1995). Since the beginning of nineties a variety of *Stochastic Process Algebras* (SPA) have been proposed as alternative approaches to performance modelling, see e.g. (Hermanns *et al.*, 1998b; Hillston, 1996; Buchholz, 1994; Bernardo and Gorrieri, 1998; Brinksma *et al.*, 1995). With such algebras, CTMCs can be built in a *compositional* way: since the inherent structure of nowadays and tomorrows systems is becoming more and more complex, the possibility to specify CTMCs in a compositional way is a significant advantage.

However, the benefits of compositionality are not restricted to a reduced effort needed to specify the model. As pointed out for instance in (Hillston, 1995), equivalence preserving transformations can be applied to reduce the analytical effort required by the model, in particular to reduce the size of the underlying state space and the corresponding CTMC.

Equivalence relations that are applied for this purpose are based on a stochastic adaptation of the notion of bisimulation, *the* central equivalence for non-stochastic process algebra (Milner, 1989). Their compositional application is particularly successful and leads to a significant reduction of the state space of the model if the system under consideration contains identical components. First, however, the symmetries have to be detected by generating the state space, or by introducing some kind of syntactic transformation. The latter approach is, for instance, applied in (Hermanns *et al.*, 1995) using equational laws; a general technique for detecting some forms of symmetries at a syntactic level is discussed in (Gilmore *et al.*, 1998).

In the context of Petri nets, similar observations have lead to the definition of Stochastic Well Formed Nets (SWN) (Chiola *et al.*, 1993) which extend GSPNs with colours as means to express symmetries at the specification level. Also uncoloured GSPNs allow a certain degree of symmetry representation, since a place may contain multiple tokens when their identity is irrelevant. In this situation the resulting state space can be much smaller than the state space of a corresponding 1-safe net.

We develop a similar concept for SPA, introducing *symmetric parallel composition*, an operator that realises n-ary parallel composition of identical replicas, strongly inspired by (RS94). Its semantics directly produces a reduced state space. Furthermore, the semantics can be proven to be correct with respect to the usual semantics of parallel composition. To explain this operator we provide an interpretation in terms of Petri nets. The paper is organised as follows: After a brief sketch of the language we introduce symmetric composition and its properties. Then we show the effect of this operator by means of a simple example. The next section presents a possible Petri net interpretation of symmetric composition. Before concluding

ing the paper with a short summary we review a few related approaches to exploit symmetries.

SYMMETRIC COMPOSITION

Before introducing the symmetric parallel composition operator we make more precise the scenario by briefly introducing the language we will focus on, a variant of the ISO standardised specification language LOTOS. It is the calculus of *Interactive Markov Chains* (Hermanns, 1998). Due to space constraints, we cannot present in detail the language and its theory; we invite the reader to see for example (Hermanns *et al.*, 1998b) for all the details.

The language $\mathcal{L}$ is built from the following operators, where a is an action, taken from a set of actions $Act \cup \{\tau\}$, and A is a subset of Act. We distinguish the action τ as an *internal*, invisible activity, while all other actions are *external*, describing some activity that is observable from the environment.

$\mathbf{stop}$	*inaction*
$a \; ; \; P$	*action prefix*
$(\lambda) \; ; \; P$	*Markovian prefix*
$P \,[]\, Q$	*choice*
$P \,\|[A]\|\, Q$	*parallel composition*
$\mathbf{hide}\ A\ \mathbf{in}\ P$	*hiding*
X	*process instantiation*

A set of process definitions (of the form $X := P$) constitutes a process environment.

The operators have the following informal meanings: $\mathbf{stop}$ represents a process that cannot perform any action. $a \; ; \; P$ executes an action a and then behaves like P. The time instant at which a is executed is not determined, except if the action is internal (i.e. $a = \tau$). In the latter case, the action is performed *immediately* (in zero time), while an external action can potentially be the subject of some external influence affecting its execution time. $(\lambda) \; ; \; P$ is the process that after a certain delay evolves to the behaviour of P. This delay is governed by a negative exponential distribution with rate λ. The expression $P \,[]\, Q$ behaves either as P or Q, but not both. The decision between these alternatives is taken in favour of the fastest, i.e. the first alternative that is able to evolve further, either because some delay has elapsed, or because some action can occur. This is known as the race condition. If this fastest process is not uniquely determined, a non-deterministic selection among the fastest processes is made. In $P \,|[A]|\, Q$ the processes independently execute all actions not belonging to the set A, and synchronise on those in the set A. Note that internal actions are not allowed to occur in any synchronisation set A. This is the reason why internal actions cannot be delayed due to the environment, while external actions can be subject to further synchronisation constraints influencing the time of their execution. Note also that, different from other SPAs the synchronisation of delays is ruled out in our language, avoiding some semantical ambiguities. The process $\mathbf{hide}\ A\ \mathbf{in}\ P$ behaves in the same way as P, except that the actions in the set A are *internal*ised, they are turned into internal actions τ.

The formal semantics of the language is defined in a structural operational style (Plotkin, 1981), defining action transitions, $\xrightarrow{a}$, and Markovian transitions, $\dashrightarrow^{\lambda}$, between expressions of the language. Since parallel composition plays the role of a yardstick for symmetric composition,we discuss here the operational semantic rules for parallel composition.

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{a} Q'}{P \,|[A]|\, Q \xrightarrow{a} P' \,|[A]|\, Q'} \quad a \in A$$

$$\frac{P \xrightarrow{a} P'}{P \,|[A]|\, Q \xrightarrow{a} P' \,|[A]|\, Q} \quad a \notin A$$

$$\frac{Q \xrightarrow{a} Q'}{P \,|[A]|\, Q \xrightarrow{a} P \,|[A]|\, Q'} \quad a \notin A$$

$$\frac{P \dashrightarrow^{\lambda} P'}{P \,|[A]|\, Q \dashrightarrow^{\lambda} P' \,|[A]|\, Q}$$

$$\frac{Q \dashrightarrow^{\lambda} Q'}{P \,|[A]|\, Q \dashrightarrow^{\lambda} P \,|[A]|\, Q'}$$

These rules are read as follows: if the transition above the line is possible, then we can infer the transition below the line, provided that the side condition holds. The first rule handles synchronisation, where both processes P and Q have to change state on the occurrence of an action a contained in the set A. The remaining four rules handle the asynchronous execution of action transitions and Markovian transitions. As a consequence of the *memoryless property* enjoyed by negative exponential distributions, Markovian transitions can be simply interleaved (without the need to adjust distributions in the idling process).

By applying the semantic rules it is possible to obtain a transition system underlying any expression, i.e. a directed graph describing its behaviour. As an example, Figure 1 shows the transition system of $P \,|[a]|\, P \,|[a]|\, P$, where $P := (\lambda) \; ; \; a \; ; \; \mathbf{stop}$.

One of the most important aspects of (stochastic) process algebras is the possibility of defining equivalence relations between processes which can be used to compare

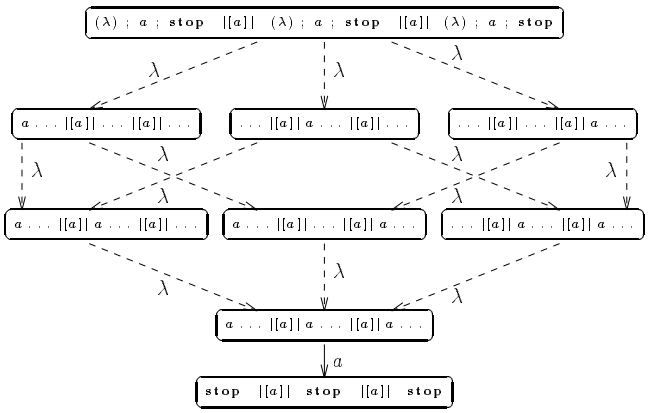


Figure 1: Parallel composition of identical processes

process specifications and to replace one specification by another one which exhibits an *equivalent* behaviour, but possibly has a different representation (such as a smaller state space). A central notion in this context is the notion of *bisimulation* (Milner, 1989) which compares processes according to the actions they are able to perform. In the essence, any action a either process is able to execute can be matched by an execution of the same action a of the other process such that the processes remain equivalent after these executions.

On the level of Markov chains, a corresponding definition is provided by the notion of *lumpability* (Kemeny and Snell, 1976; Hillston, 1996). By imposing constraints on actions and rates, *strong Markovian bisimilarity* reflects lumpability *and* bisimulation on Markovian transitions, respectively action transitions (Hermanns *et al.*, 1998b). A corser notion of equivalence, *weak* Markovian bisimilarity, additionally allows abstraction from internal computations, in the spirit of (Milner, 1989). For this purpose, sequences of internal actions are treated as being irrelevant since they do not consume time and are not observable. The details can be found in (Hermanns, 1998).

Let us now introduce the symmetric composition operator and its semantics. Parallel composition of identical components usually causes unnecessary growth of the state space, if the traditional operator $|[A]|$ is used for this purpose. For instance, in the transition system of Figure 1, the states in the second (as well as in the third) row are equivalent, since they are just permutations of each other. The symmetric composition of n replicas of P is denoted by $\{n \,!\, P\}A$ to indicate that n identical components evolve independently by spending time or performing actions not belonging to the synchronisation set A; actions contained in the set A, instead, have to be performed synchronously by *all* replicas.

Consider, for example, the transition system of Figure 2. The process $\{3 \,!\, (\lambda)\,;\, a\,;\, \mathbf{stop}\}\{a\}$ will first spend some time, since the three entities enable Markovian delays. The time until it changes to another behaviour is given by the minimum[1] of three exponential distributions, all with rate λ, i.e. the rate until a change happens is 3λ. Then, one of the replicas will change its behaviour to $a \;;\; \mathbf{stop}$ while the other two will stay in their current state. We denote this situation by the term $\{2 \,!\, (\lambda)\,;\, a\,;\, \mathbf{stop}\,,\, 1 \,!\, a\,;\, \mathbf{stop}\}\{a\}$. Note that this term contains two different expressions with their respective multiplicities. What is the subsequent behaviour of this term? Performing a is not possible, because only one out of three replicas is ready to perform it. Therefore, in complete analogy to the above, this process will evolve to $\{1 \,!\, (\lambda)\,;\, a\,;\, \mathbf{stop}\,,\, 2 \,!\, a\,;\, \mathbf{stop}\}\{a\}$ with rate 2λ. This process is still not able to perform a and it will hence evolve to $\{3 \,!\, a\,;\, \mathbf{stop}\}\{a\}$ with rate λ. Now, all processes are ready to perform a, leading to the final state $\{3 \,!\, \mathbf{stop}\}\{a\}$.

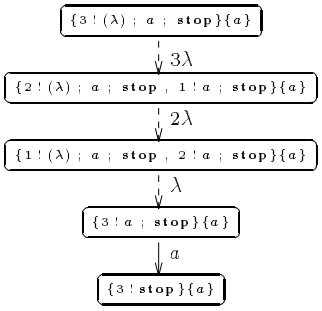


Figure 2: Symmetric composition of processes

This simple example illustrates the potential benefits of an operator to explicitly specify such symmetric behaviours. In order to formalise the semantics of this operator, we essentially deal with multisets of expressions. The bridge between the notation $\mathcal{M} = \{n_1 \,!\, P_1, \ldots, n_m \,!\, P_m\}$ and a standard multiset over $\mathcal{L}$, i.e. a function (also denoted $\mathcal{M}$) from $\mathcal{L}$ to $\mathbb{N}$ is the following definition:

$$\mathcal{M}(P) = k > 0 \quad \text{if and only if} \quad (k \,!\, P) \in \mathcal{M}.$$

We use a common multiset operation to manipulate multisets, namely insertion ($\oplus$) of elements into a multiset. It is defined as $\mathcal{M} \oplus P := \mathcal{M}'$, where

$$\mathcal{M}'(P') := \text{ if } P' \equiv P \text{ then } \mathcal{M}(P') + 1 \text{ else } \mathcal{M}(P')$$

After these notational preliminaries we are ready to define the semantics of our new operator. It is given by the following operational rules:

[1] The minimum of a set of exponential distributions is given by an exponential distribution with a rate equal to the sum of the individual rates.

$$\frac{}{\emptyset A \xrightarrow{a} \emptyset A} \quad a \in A$$

$$\frac{\mathcal{M}A \xrightarrow{a} \mathcal{M}'A \qquad P \xrightarrow{a} P'}{(\mathcal{M} \oplus P)A \xrightarrow{a} (\mathcal{M}' \oplus P')A} \quad a \in A$$

$$\frac{P \xrightarrow{a} P'}{(\mathcal{M} \oplus P)A \xrightarrow{a} (\mathcal{M} \oplus P')A} \quad a \notin A$$

$$\frac{P \xrightarrow{\lambda} P'}{(\mathcal{M} \oplus P)A \xrightarrow{n\lambda} (\mathcal{M} \oplus P')A} \quad \mathcal{M}(P) = n - 1$$

The first two rules handle synchronisation, where all replicas have to change state on the occurrence of an action contained in the set A. The necessary preconditions that all elements in the multiset are able to perform an action are checked element-wise, until the remaining set is empty. The latter two rules of symmetric composition handle asynchronous action transitions, respectively Markovian transitions, in a straightforward way; notice that the multiplicity of enabled Markovian delays is taken into account in the rate $n\lambda$ in the last rule.

A central property of symmetric composition is its consistency with traditional parallel composition. The proof proceeds by induction on n, but is omitted due to space constraints.

Theorem For all $n > 0$, we have that $\{n\,!\,P\}A$ is strong Markovian bisimilar to $\underbrace{P \,||[A]|\, \ldots \,||[A]|\, P}_{n \text{ times}}$.

Symmetric composition always produces a smaller, but bisimilar, transition system compared to the traditional notation.

In the next section we will show by example that symmetric composition can turn an exponential blow up (in the number of parallel components) into a quadratic growth, if the components are identical replicas. Indeed, for that example, the state space produced by symmetric composition is *minimal*. A transition system is minimal if it possesses the least possible number of states necessary to represent a certain equivalence class of behaviours. This raises the question whether symmetric composition *always* generates the minimal transition system, if we choose strong Markovian bisimilarity as the relevant equivalence. The experiences we sketched so far seem to suggest this desirable property. However, minimality does not hold in general, but can be formally shown for a restrictied class of expressions, *linear* processes, that neither involve choice nor parallel (or symmetric) composition. It appears feasible to extend this property beyond linear processes, but it is not obvious how to syntactically characterise a larger class such that, for instance, the example given in the next section is included.

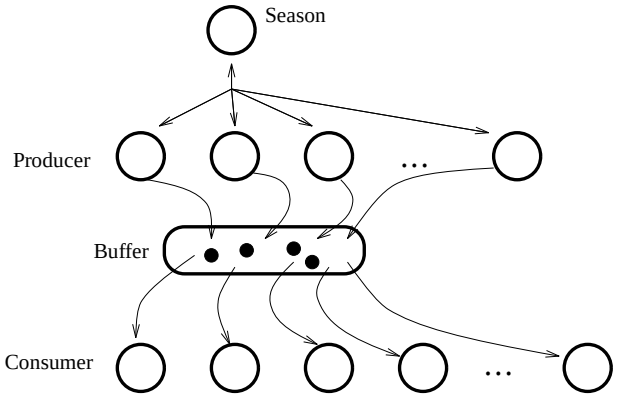


Figure 3: A simple producer/consumer example

REDUCTION BY EXAMPLE

In this section we study the benefits of symmetric composition by means of an example. In particular, we compare the growth of the state space when the model is specified using the symmetric and the traditional parallel composition operators. We investigate a simple producer/consumer example, parametric in the number of producers and consumers, whose structure is depicted in Figure 3. Each producer generates jobs and then delivers them to a buffer, common for all producers. If we restrict the buffer size to five places, a possible specification is:

$$\begin{aligned} Buffer_0 &:= put; Buffer_1 \\ Buffer_i &:= put; Buffer_{i+1} \,[]\, get; Buffer_{i-1} \\ &\qquad\qquad i = 1,2,3,4 \\ Buffer_5 &:= get; Buffer_4 \end{aligned}$$

The producers individual rate for generation of jobs alternates between two different values. In a high load phase the rate is smaller, i.e. jobs are generated more often than in a low load phase. The change of phase is initiated by an external signal c.

$$\begin{aligned} Producer &:= (high); put; Producer \,[]\, c; ProdLow \\ ProdLow &:= (low); put; ProdLow \,[]\, c; Producer \end{aligned}$$

This external signal c is periodically emitted by a (rather slow) synchronising process.

$$Season := (slow); c; Season$$

The consumers simply take jobs out of the buffer and work on them (with a certain rate).

$$Consumer := get; (work); Consumer$$

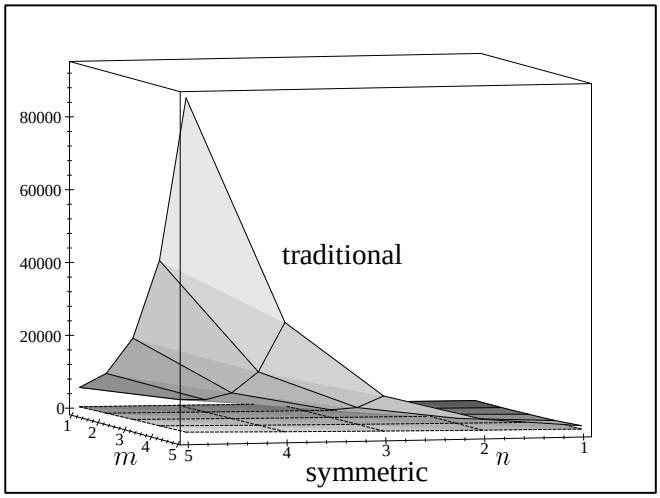


Figure 4: Size of the transition system

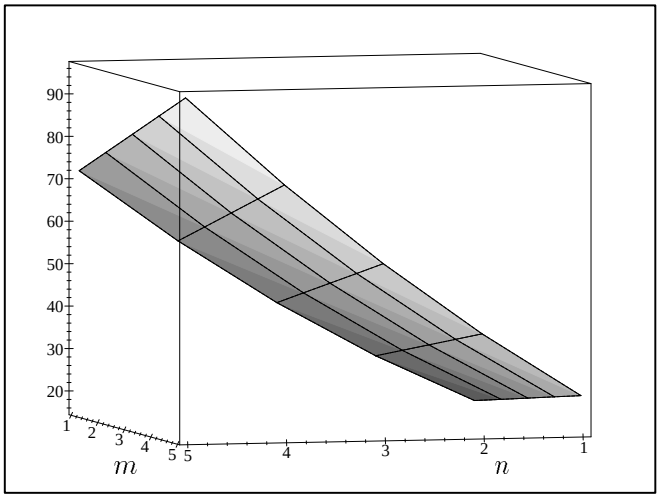


Figure 5: Size of the lumped Markov chain

The overall structure is as follows:

$$System := \begin{array}{l} \mathbf{hide}\, c, put, get\, \mathbf{in} \\ (Season \;|[c]|\; Producers_n) \;|[put]| \\ Buffer_0 \;|[get]|\; Consumers_m \end{array}$$

where the producers and the consumers are parametric. Using traditional parallel composition they may be substituted by:

$$Producers_n := \underbrace{Producer|[c]|\ldots|[c]|Producer}_{n \text{ times}}$$

$$Consumers_m := \underbrace{Consumer|||\ldots|||Consumer}_{m \text{ times}}$$

In the sequel, we will refer to this specification as the *traditional* specification. Note that $Consumer$ replicas are completely independent from each other, whereas $Producers$ have to synchronise on c. On the occurrence of this signal, all of them synchronously change from one load phase to another. As a consequence, the production of jobs is a certain Markov modulated Poisson process.
Using symmetric composition (and the theorem of the previous section), the whole specification can be equivalently expressed by the following expression, that we will call *symmetric* in the sequel.

$$\begin{array}{l} \mathbf{hide}\, c, put, get\, \mathbf{in} \\ (Season \;|[c]|\; \{n \,!\, Producer\}\{c\}) \;|[put]| \\ Buffer_0 \;|[get]|\; \{m \,!\, Consumer\}\emptyset \end{array}$$

We have constructed the transition system underlying both specifications with the TIPPtool (Hermanns *et al.*, 1998a).

Figure 4 compares the traditional description with the one where symmetric parallel composition is used. The traditional description grows truly exponential in both parameters, from 72 states ($n = 1, m = 1$) to 93312 states ($n = 5, m = 5$). In contrast, using symmetric parallel composition leads to a quadratic growth of the state space, from 72 states to only 1512 states. Note that both transition systems are strongly Markovian bisimilar, as a consequence of our theorem. In addition, the transition system generated by symmetric composition is minimal, since there is no smaller, yet (strongly Markovian) bisimilar transition system. To verify this, we have used the algorithms provided by the TIPPtool. Indeed, with these algorithms, both specifications can be compressed further, if the notion of weak Markovian bisimilarity is applied instead. As already mentioned, weak Markovian bisimilarity allows abstraction from internal activities. Since we have used hiding to internalise synchronisation of components, the application of weak Markovian bisimilarity is quite fruitful. In particular, the resulting transition system does not possess action transitions and therefore directly corresponds to a lumped Markov chain. The size of this Markov chain is depicted in Figure 5, it grows quadratic from 16 to 96 states. Note that both specifications result in the same lumped Markov chain, as an indirect consequence of the above theorem (using the fact that strong implies weak Markovian bisimilarity).
Figure 6 again compares the state space requirements of both specifications, but now *compositional aggregation* is applied to both. Some explanations are necessary to assess this plot. Compositional aggregation exploits substitutivity of language operators (in particular parallel composition) with respect to an equivalence. (Hillston, 1995) gives a good insight into the flavour of compositional ag-

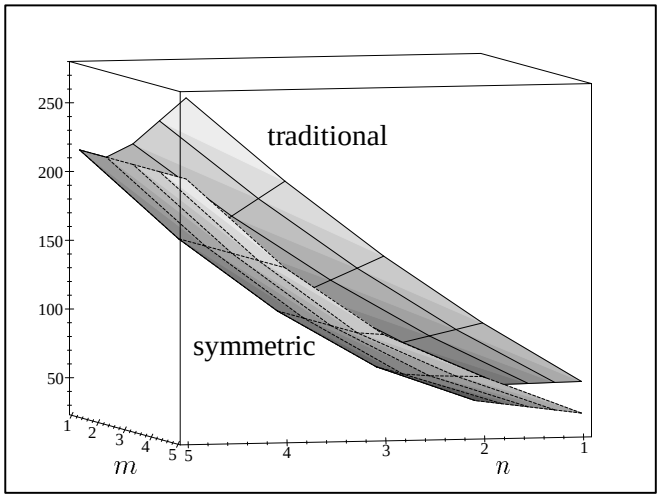


Figure 6: State space size for compositional aggregation

gregation in the framework of SPA. The general idea is to generate the state space in a stepwise fashion along the parallel structure of the specification and to minimise the state space in every step. Compositional aggregation with respect to weak bisimilarities particularly allows one to exploit abstraction from internal details. This can drastically reduce the size of the state space, as reported e.g. in (Chehaivbar *et al.*, 1996).

The interplay of hiding and parallel composition in each step allows a simple rule of thumb to achieve optimal aggregation. In order to reduce as much and as early as possible, it is wise to internalise actions that are local for certain components. After incorporating this rule of thumb into our specification, we have performed compositional aggregation with respect to weak Markovian bisimilarity for both types of our system specification, again varying n and m between 1 and 5. With compositional aggregation, it makes no sense to compare the result of the whole aggregation procedure, since the result of the very last aggregation step is always minimal. In other words, compositional aggregation for the traditional as well as the symmetric specification produces the results depicted in Figure 5. A proper comparison is based on the maximum number of (intermediate) states that have to be stored during compositional aggregation. This is what is depicted in Figure 6. In this comparison, symmetric composition (28 to 216) is still, though only slightly, better than traditional composition (28 to 275). Note that an increasing m does not always affect the maximum usage of state space. This is particularly true for the symmetry exploiting specification, but also for the traditional description (for small m). It is caused by the fact that m comes into play in the very last composition step(s), but former composition steps have already led to larger intermediate state spaces.

PETRI NET INTERPRETATION

In this section we give an intuitive interpretation of symmetric composition in terms of Petri nets[2]. In (Bernardo *et al.*, 1995b; Ribaudo, 1995) net semantics for SPAs have been proposed: in both cases the nets which are obtained are 1-safe, i.e. each place can contain at most one token. (Bernardo *et al.*, 1995a) proposes a label-oriented net semantics in which the restriction to 1-safeness is relaxed in order to obtain more compact models in which places can hold many tokens.

Symmetric composition $\{n \,!\, P\}A$ specifies that n replicas of P cooperate on the actions appearing in the set A and otherwise proceed independently. The question is: what is the Petri net counterpart for this operator?

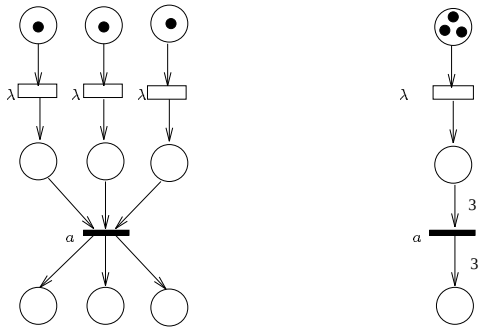
Let us first consider the term $\{n \,!\, P\}\emptyset$: starting from a net representation of P, we can easily interpret the number n as the number of tokens forming the initial marking of this net structure. In this sense, symmetric composition approaches the notion of places with multiple tokens at the level of the process algebra specification.

Let us now turn our attention to symmetric composition with synchronisation of all replicas, as in $\{n \,!\, P\}A$. Usually the synchronisation between activities is translated considering the superposition of transitions with the same label occurring in the synchronisation set. Since we know the number of identical replicas and that they all synchronise on the same actions belonging to A, we can build the net of a single component and then model the synchronisations by associating proper multiplicities with the input and output arcs of the transitions modelling the synchronising activities. These multiplicities are given again by the number n appearing in $\{n \,!\, P\}A$.

Returning to the first example discussed when introducing symmetric composition, the two nets shown in Figure 7 corresponds to the translations of the terms $P \,||[a]||\, P \,||[a]||\, P$ and $\{3 \,!\, P\}\{a\}$ (assuming that timed transitions with infinite server semantics model Markovian delays and immediate transitions model immediate actions). The GSPN in Figure 7(a) is obtained by applying a variant of the net semantics defined in (Ribaudo, 1995) for the traditional composition of identical processes. The GSPN in Figure 7(b) instead translates symmetric composition by applying the recipe discussed above. The reachability graphs underlying these two nets are isomorphic to the transition systems shown in Figures 1 and 2.

The mapping of the symmetric composition operator onto GSPN models becomes more complicated if the algebraic specification of the single components involves more than

[2]We are indeed considering GSPNs since our language has both Markovian delays and immediate actions.



(a) (b)

Figure 7: Petri net interpretation of traditional and symmetric composition

a single action labelled with a name in the set A. This is the case, for instance, in the expression $\{2\,!\,Q\}\{a\}$ where $Q := (\lambda)\,;\,(a\,[]\,b)\,[]\,(\lambda)\,;\,(a\,[]\,c)$, whose resulting GSPN model is shown in Figure 8.

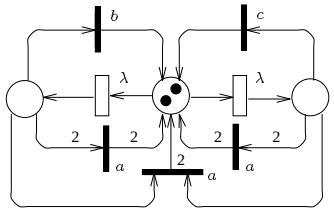


Figure 8: GSPN model of $\{2\,!\,Q\}\{a\}$

With the introduction of symmetric composition, we abstract from some information about the modelled system. The operator relies on the fact that it is not possible to distinguish which replica is actually performing some action just by observing the behaviour. This behaviour oriented view is essential for process algebras and indeed the basic justification for the theorem introduced earlier. In the Petri net formalism a similar view is taken implicitly whenever identical subnets are folded together in order to obtain a more compact model. However, the effect of symmetric composition is different from the one introduced in the context of SWNs where a more compact representation is obtained by adding a colour structure at the net level. In fact, our symmetric composition operator can be mapped onto an uncoloured net since the behaviour oriented view only considers situations in which the identity of the components is not significant.

It is worth noticing that symmetric composition breaks a limitation of SPAs compared to GSPNs: A single Petri net structure can be used to represent several instances of the same system by changing its initial marking. So far, this possibility was provided by the SPA formalism only by explicitly enumerating the components at the specification level. Symmetric composition instead is parametric in n and a change of n matches a change in the initial marking of the corresponding net. The Petri net interpretation of symmetric composition mainly highlights that this operator incorporates a notion of *token multiplicities* into the context of SPA.

RELATED WORK

Some other techniques have been proposed to generate a reduced state space in the context of SPA. (Hermanns *et al.*, 1995) uses equational laws to perform syntactic rewriting of expressions that are symmetric. Although that paper only exploits strong bisimilarity on a purely Markovian calculus, i.e. where actions are linked to delays, this technique is equally possible for the calculus presented here, using the equational characterisation of (Hermanns, 1998). A comparison of such a technique with the ones discussed here is difficult. First, syntactic rewriting does not produce a state space. Instead, it shrinks and expands the syntactic specification by applying rewrite rules. In addition, different ordering of rewrite rule applications may lead to drastically different storage requirements. However, for the special case we considered here, where identical replicas of components are tackled, it seems to be not too difficult to develop a high level transformation law that produces the same result on the syntactic level that our operator does on the level of transition systems.

An approach in a similar direction has recently been developed in (Gilmore *et al.*, 1998), yet restricted to a purely Markovian calculus. In that paper a reduced state space for symmetric systems is generated by storing only one among a set of equivalent processes. The proposed technique also works at the level of syntactic descriptions: processes are considered to be equivalent if they can be obtained by permutation of symmetric components within parallel composition and hiding operators. For each set of equivalent processes the one which is minimal with respect to the lexicographic ordering is considered as the representative of the set and hence explicitly stored in the reduced state space. Before the generation of the reduced state space, a pre-processing phase is necessary for syntactic rearrangements of the model. These transformations, however, are able to handle symmetries beyond the power of symmetric composition. For instance, they exploit symmetries also if replicas are synchronised via different synchronisation sets, as in $(P\,|[a]|\,P)\,|[b]|\,P$. This is not possible with symmetric composition via $\{n\,!\,P\}A$,

because the synchronisation set is fixed for all replicas.

CONCLUSION

In this paper we have introduced symmetric composition for SPA; its operational semantics is compact and intuitive, and consistent with the traditional operator. We have also provided an informal Petri net description of this operator. It highlights that symmetric composition incorporates a notion of token multiplicities into SPA.

By means of an example we have shown that the traditionally observed exponential growth of the state space can be turned into an at most quadratic growth, and we have also illustrated the beneficial effect of compositional aggregation on this example.

The transition system generated by symmetric composition is minimal (with respect to strong Markovian bisimilarity) in many cases, even though we have only been able to prove this property for the class of linear processes. Future work will study how this result can be extended to more general classes of processes. In order to achieve this, it seems helpful to provide a formal definition of the Petri net semantics of symmetric composition.

Another point for future work is the relation to other approaches for the generation of reduced state spaces for SPA models; for instance, it is worth to be investigated how the operator can be extended into the direction of (Gilmore *et al.*, 1998) where only a part of the replicas can synchronise on certain actions.

The language used in this paper deviates from other SPAs since it does not rely on synchronisation of Markovian transitions. In this way, we have avoided the need to provide an intuitive stochastic interpretation of the rate to be associated with a synchronising Markovian transition. All other approaches have to address this point, but they provide rather different interpretations. However, this decision is orthogonal to the definition of symmetric parallel composition and we claim that any kind of substitutive parallel composition of Markovian transitions can be easily adopted to our symmetric composition.

REFERENCES

Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. 1995. *Modelling with Generalized Stochastic Petri Nets.* Wiley.

Bernardo, M., and Gorrieri, R. 1998. "A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time". *Theoretical Computer Science*. to appear.

Bernardo, M., Busi, N., and Gorrieri, R. 1995a. "A Distributed Semantics for EMPA based on Contextual Nets". *The Computer Journal*, **38**(7).

Bernardo, M., Donatiello, L., and Gorrieri, R. 1995b. "Giving a Net Semantics to Markovian Process Algebras". *In: Proc. 6th Int. PNPM Workshop.*

Brinksma, H., Katoen, J.-P., Langerak, R., and Latella, D. 1995. "A stochastic causality-based process algebra". *The Computer Journal*, **38**(7).

Buchholz, P. 1994. "Markovian Process Algebra: Composition and Equivalence". *In: Proc. 2nd PAPM Workshop*, Arbeitsberichte des IMMD 27(4).

Chehaivbar, G., Garavel, H., Tawbi, N., and Zulian, F. 1996. "Specification and Verification of the Powerscale Bus Arbitration Protocol: An Industrial Experiment with LOTOS". *In: Formal Description Techniques IX*. Chapmann Hall.

Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. 1993. "Stochastic Well-Formed Coloured Nets for Symmetric Modelling Application". *IEEE Transaction on Computers*, **42**(11).

Gilmore, S., Hillston, J., and Ribaudo, M. 1998. *"An Efficient Algorithm for Aggregating PEPA Models"*. submitted for publication.

Hermanns, H. 1998. *Interactive Markov Chains*. Ph.D. thesis, Universität Erlangen-Nürnberg.

Hermanns, H., Herzog, U., and Mertsiotakis, V. 1995. "Stochastic Process Algebras as a Tool for Performance and Dependability Modelling". *In: Proc. of IPDS'95 Symposium.*

Hermanns, H., Herzog, U., Klehmet, U., M.Siegle, and Mertsiotakis, V. 1998a. "Compositional Performance Modelling with the TIPPtool". *In: 10th Int. Conference on Modelling Techniques and Tools for Computer Performance Evaluation.* LNCS. Springer.

Hermanns, H., Herzog, U., and Mertsiotakis, V. 1998b. "Stochastic Process Algebras - Between LOTOS and Markov Chains". *Computer Networks and ISDN Systems*, **30** (9-10).

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

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

Kemeny, J.G., and Snell, J.L. 1976. *Finite Markov Chains*. Springer.

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

Plotkin, G.D. 1981. *A Structured Approach to Operational Semantics*. Tech. rept. DAIMI FM-19. Aarhus University.

M. Rettelbach and M. Siegle. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. *In: Proc. 2nd PAPM Workshop*, Arbeitsberichte des IMMD 27(4).

Ribaudo, M. 1995. "Stochastic Petri Nets Semantics for Stochastic Process Algebras". *In: Proc. 6th Int. PNPM Workshop.*