Compositional Minimal Semantics for the Stochastic Process Algebra TIPP

Michael Rettelbach, Markus Siegle

Abstract

The problem of deriving the Markov chain underlying a stochastic process algebra term is addressed. Transition rate matrices are used as a convenient method for uniquely describing Markov chains. For a modified version of the stochastic process algebra TIPP, we propose a set of new semantic rules which specify the way in which process terms are translated into their corresponding matrices. For each operator of the language, a semantic rule describes how the (one ore more) operand matrices have to be combined in order to form the matrix corresponding to the overall term. These semantic rules guarantee certain highly advantageous properties of the resulting matrices, the two most important of which are (i) the absence of non-reachable states and (ii) minimality with respect to Markov chain lumpability. Thus avoiding redundancy, our new approach is a contribution to the struggle against state space explosion.

1 Introduction

TIPP is a formal language for specifying Markovian models of behaviour, employing a process algebra formalism. For a general introduction to TIPP see [GHR93, Göt94]. This paper deals with the problem of translating a TIPP description into its underlying Markov chain whose subsequent solution permits the calculation of performance measures for the model. The major difficulty lies in the potential size of the resulting Markov chain which in many cases can make its numerical solution impractical. On the other hand, Markov chains resulting from TIPP descriptions often have subsets of states which represent equivalent behaviour, such that their individual treatment is redundant. Therefore we are interested in the generation of Markov chains which are *minimal*, i.e. do not have states with equivalent behaviour.

One can think of the following three general schemes for translating a TIPP term into a labelled transition system which is minimal with respect to a given notion of equivalence:

- In a first step translating the term into a labelled transition system following the ordinary structural operational TIPP semantics. Afterwards the transition

system is checked for sets of equivalent states. Given the existence of such sets the transition system can be transformed into an equivalent but smaller one.

- Rewriting the term with respect to the rules of an axiomatisation [HR94], thereby obtaining a normal form of the process term, and applying the structural operational semantics afterwards.
- Applying improved semantic rules which produce *directly* the reduced state space.

The new approach proposed in this paper follows the third scheme. It defines a compositional semantics which guarantees in each step the generation of a minimal Markov chain. We do not regard labelled transition systems but work on a related formalism — the matrix representation of the underlying Markov chain.

2 Preliminaries

2.1 The Language

We define a new language, TIPP^{MS}, which is based on a subset of TIPP, enhanced by a new replication operator. This language is defined by the following grammar:

$$P ::= 0 \quad | \quad X \quad | \quad (a, \lambda).P \quad | \quad P + P \quad | \quad recX.P \quad | \quad !_S^n P$$

where $a \in Act$ and $X \in Var$ is a process variable. For the moment, we restrict the set of action types to one single action, i.e. $Act = \{a\}$. This restriction permits a simpler notation and explanation of the semantic rules which will be established in Sec. 3. Thereafter, the generalisation to more than a single action type will turn out to be straight forward.

The operators of the language have the usual meaning (cf. e.g. [GHR93]). The new replication operator $!_S^n$ is defined as:

$$!_S^n A := \underbrace{A \parallel_S A \parallel_S \ \ldots \ \parallel_S A}_{n \text{ times}}$$

Thus, the replication operator $!_S^n$ is a specialisation of the general parallel composition operator $\parallel_S$ which we have not included in the language TIPP^{MS}. The language TIPP^{MS} also does not provide a hiding operator.

2.2 Equivalence and Lumpability

Equivalent behaviour and bisimulation In an algebra it is important to provide a means for comparing terms, in particular to establish the notion of equality between two terms. A whole line of research has been directed towards the definition of equivalence relations for process algebras. For process algebras in a Markovian context the major contributions are [Hil94], [HR94] and [Buc94b].

In the following, we will talk on the one hand of the equivalence between two process terms, and on the other hand of the equivalence between two states. Since states are in fact (derived) process terms, the meaning is the same in both cases. Informally, two states of a labelled transition system (two process terms) are *equivalent* if and only if for all actions the overall rate to change the state to any state of a given equivalence class is the same for both processes and for all equivalence classes. For a more detailed explanation and a formal definition of the notion of equivalence that is used within this paper, please see [HR94].

Having established the notion of equivalence, it is natural to look for a *minimal* representation of a state, i.e. an equivalent and most concise description of a state. This goal is very closely related to our goal in the work presented here: To derive a minimal matrix representation of the behaviour specified by a process term.

Relation between MPA equivalence and CTMC lumpability In [Hil93], Hillston showed that the stochastic process underlying a Markovian process algebra (MPA) is a continuous time Markov chain (CTMC). A convenient way for unambiguously specifying a CTMC is its description by a transition rate matrix. There exists a notion of minimality in the CTMC context which is induced by the concept of CTMC lumpability.

Let us regard a partition $\Omega = \{\omega_1, \omega_2, \ldots\}$ of the state space of a CTMC. The CTMC is ordinarily lumpable [KS76, Buc94a] with respect to this partition if for any two states $i, j \in \omega_I$ from the same subset ω_I the total rate to another subset ω_J is the same:

$$\forall I : \forall i, j \in \omega_I : \forall J : r_{iJ} = r_{jJ} \quad \text{where } r_{iJ} = \sum_{j \in J} r_{ij}.$$

Here, r_{ij} denotes the transition rate from state i to state j. On the matrix level this condition can be conveniently expressed by a partial row sum criterion.

The measures of an original CTMC and its lumped counterpart are strongly related. The (macro-)probability of the lumped Markov chain being in state I is equal to the sum of the associated (micro-)probabilities:

$$\forall I : p(I) = \sum_{i \in \omega_I} p(i)$$

This is true for both transient and stationary probabilities.

If a CTMC is not further lumpable we call it *minimal*. This existence of a minimality notion for CTMCs justifies the fact that we focus on the matrix world and use lumpability arguments when deriving minimal representations for process algebra terms. The relation between process algebra equivalence and Markov chain lumpability has been noticed previously by [Hil94] and [Buc94b].

3 Compositional Minimal Semantics

In this section we will formulate semantic rules for the operators of the language TIPP^{MS}. The idea is to define a direct mapping from process terms to transition rate matrices. The latter unambiguously specify the Markov chain underlying a TIPP^{MS}-description.

If A is a TIPP^{MS}-term, then the semantics $Q_A = (a_{ij}) = [\![A]\!]_{MS}$ of A is an $n \times n$ matrix where $a_{ij} \in R^+ \cup Var$. The entry a_{ij} denotes the rate of transition from state i to state j due to an action, whose type — for the moment — is implicitly known to be the single action type a. In particular, self-loops are allowed, which means that diagonal entries a_{ii} are also non-negative reals. We do *not* use infinitesimal generator type matrices (whose diagonal entries equal the negative row sums).

We will impose the following conditions on the type of matrices we use for describing process behaviour:

1. In order to avoid waste of memory space for unreachable states (which do not contribute to the behaviour anyway) we impose the restriction that all states of the matrix have to be *reachable* from the first state.

2. For the reasons explained in Sec. 1 we require the matrices to be *minimal* in the sense that there are no subsets of equivalent states in the matrix.

3. By convention, the first state has the special meaning to be the *initial state* of the behaviour, i.e. the starting point.

We will refer to these three conditions as R, M and I, and call the respective matrices of RMI-type.

To start with, for the terminated process 0 and the process term consisting of only a single process variable X we trivially obtain the following matrix semantics:

$$\begin{aligned} [\![0]\!]_{MS} &:= (0) \quad (1 \times 1 - matrix) \\ [\![X]\!]_{MS} &:= (X) \quad (1 \times 1 - matrix) \end{aligned}$$

3.1 Prefixing

Let us suppose that the semantics of the process term A is described by the matrix Q_A which is already known. We are interested in the matrix semantics of the process term $(a, \lambda).A$. Prefixing usually changes the initial state of a process by adding a new initial state and making the old initial state the (single) successor of the new initial state. If one looks at this from a matrix perspective, one notices that prefixing with (a, λ) attaches a vector

$$(0 \quad \lambda \quad 0 \quad \ldots \quad 0)$$

to the top of the matrix Q_A (see Fig. 1).

$$[\![(a,\lambda).A]\!]_{MS} = \begin{array}{|c|ccc|}\hline 0 & \lambda & 0 & \ldots \quad 0 \\ \hline 0 & & & \\ \vdots & & Q_A & \\ 0 & & & \\ \hline \end{array}$$

Figure 1: Prefixing regular

It is, however, possible that our demand for minimality of the matrix is not fulfilled. This is the case if and only if there already exists a state within the matrix Q_A, from where only a single transition is possible, namely to the old initial state with the same rate λ (see Fig. 2). Such a state and the new starting state would be equivalent.

$$Q_A = \begin{array}{|ccc|}\hline & & \\ & & \\ \hline \lambda & 0 \ldots & 0 \\ \hline & & \\ & & \\ \hline \end{array} \qquad Q_{(a,\lambda).A} = \begin{array}{|c|ccc|}\hline 0 & \lambda & 0 \ldots & 0 \\ \hline & & & \\ & & & \\ \hline & & & \\ & & & \\ \hline \end{array}$$

Figure 2: Prefixing special

This special situation can be recognised very easily through the existence of a row (say number i) of the form

$$(\lambda \quad 0 \quad \ldots \quad 0)$$

within the matrix Q_A. In this case, as a consequence of the prefixing with (a,λ), state i will become the new starting state, so by convention it has to be made the first state of the matrix. This can be achieved by permuting the rows and columns of the matrix, so that the former number i is now number 1, and all positions $1 \ldots i-1$ get increased to $2 \ldots i$ (see Fig. 2).

3.2 Choice

The behaviour of a process $A + B$ is easily described: The first transition can be any transition possible either for process A or for process B. The first transition taken determines whether the process will in the sequel behave as either process A or process B. Consider the two matrices Q_A and Q_B (cf. Fig. 3). Their first rows are denoted q_A and q_B, respectively. The matrix resulting from the operation $[\![+]\!]$ is sketched in Fig. 4.

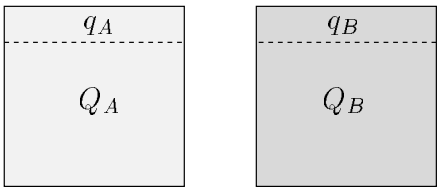

Figure 3: Matrices Q_A and Q_B

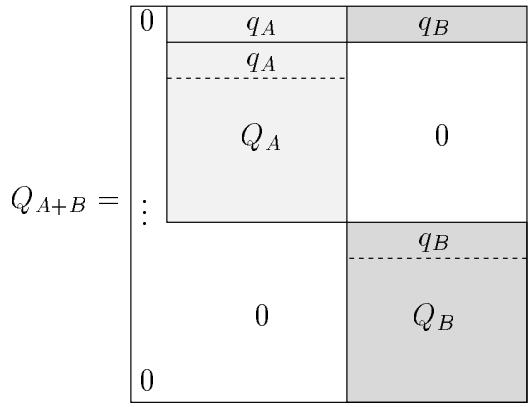

Figure 4: Matrix Q_{A+B}

The possible transitions from the two (old) initial states (rows q_A and q_B) are copied and pasted together (with a leading 0) to define the transitions which are possible from the new initial state

$$(0 \quad q_A \quad q_B) \qquad .$$

From this initial state the system can either reach the matrix Q_A or Q_B. Once it has chosen one of both, there is no chance to get back again to the initial state or to the other submatrix.

This matrix describes completely the behaviour of the process $A + B$ but unfortunately does not necessarily fulfil the RMI-conditions established at the beginning of this Section, due to the two following reasons.

First, considering condition R, one finds that some states might not be reachable any more. As long as the matrices Q_A and Q_B were of RMI-type, this can only happen to the two former initial states. This situation is easy to recognise, because for a state i to be reachable, the ith column must contain at least one non-zero element. Therefore, if the first column of Q_A (Q_B) contains only zero elements, the state which corresponds to the initial state of A (B) has to be deleted.

The second reason for the possible lack of RMI-property is the fact that both processes A and B might engage in some common patterns of behaviour. As a consequence,

constructing the matrix for $A+B$ in the manner explained above will result in a matrix with equivalent states, i.e. condition M is violated.

This second case is more complicated and cannot be recognised easily. In order to cope with this problem we define a *normal form* of a transition rate matrix Q which simplifies the recognition of common behaviour of two processes. For a motivation of this normal form, see Fig. 5 which shows the general form of a transition system.

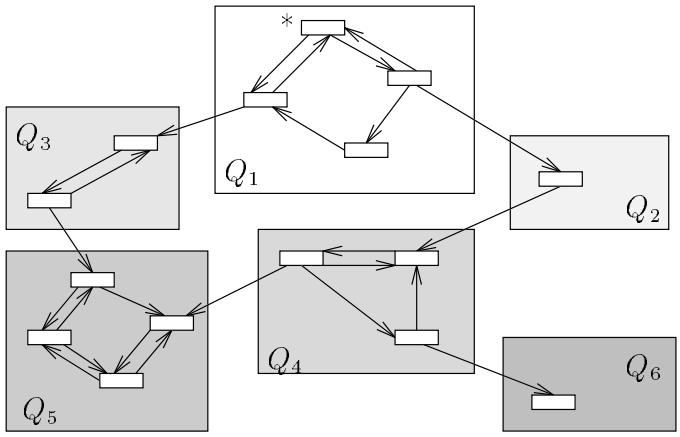

Figure 5: General form of a transition system

It can be observed from the Figure that the whole state space can be divided into *islands*, where the notion of an island is defined as follows: Within an island all states are mutually reachable, but once the system has left an island it will never return to the same island. As a consequence, islands can consist of a single (transient or absorbing) state, or of several states. Islands itself may be transient or absorbing.

According to the notion of an island we can organise a transition rate matrix Q as follows, in order to put it into normal form. States belonging to the same island are located in consecutive rows. The initial island is the one containing the initial state. The islands occur in the matrix in such an order, that there is no transition from island j to island i if $j > i$ (cf. Fig. 6). Such an ordering of islands can always be found, because otherwise there would exist two mutually reachable states i and j from different islands, which would contradict the condition on an island.

It is also important to note that the normal form of a transition rate matrix is not uniquely defined because of two reasons:

- More than one possible ordering of the islands may exist (for example, in Fig. 5 the islands Q_2 and Q_3 may be exchanged).
- The ordering of states within each of the islands is not fixed. In fact, this ordering is completely arbitrary, with the exception that in the first island the initial state must be in the first position.

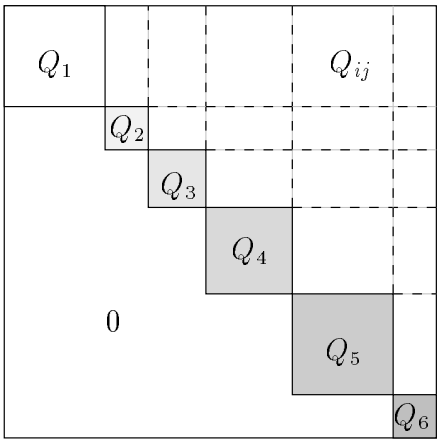

Figure 6: Normal form of a transition rate matrix Q

In the sequel we need the notion of a permutation matrix: An $n \times n$-matrix P with exactly one 1-entry in each row and in each column and all other entries equal to zero is called a *permutation matrix*. The multiplication PQ causes a permutation of the rows of matrix Q, while the multiplication QP^T causes the corresponding permutation of columns.

Using the normal form of the matrices based on their island structure makes it easier to recognise common behaviour. Assuming that both matrices Q_A and Q_B are of RMI-type, the crucial point is that a state i of Q_A represents the same behaviour as a state j of Q_B if and only if the following three conditions hold:

1. There is a permutation matrix P_1 such that $Q_i^A = P_1 Q_j^B P_1^T$, where Q_i^A is the island of Q_A containing i and Q_j^B is the island of Q_B containing j.

2. For each island $Q_{i'}^A$ of Q_A that is reachable from Q_i^A there exists a corresponding island $Q_{j'}^B$ reachable from Q_j^B which describes the same behaviour as $Q_{i'}^A$, i.e. $Q_{i'}^A = P_2 Q_{j'}^B P_2^T$.

3. The transitions from Q_i^A to $Q_{i'}^A$ correspond to the transitions from Q_j^B to $Q_{j'}^B$, i.e. $Q_{ii'}^A = P_1 Q_{jj'}^B P_2^T$.

These conditions are not as complicated as it seems from the first view. The existence of common behaviour within matrices Q_A and Q_B, both of normal form, can be checked easily in the following fashion: Starting at the absorbing islands one has to look for pairs of islands with equivalent behaviour. This only requires a test whether one of them is a permutation of the other, because absorbing islands have no successor islands. If all pairs of absorbing islands are mutually different from each other, i.e. if there is no pair of absorbing islands with equivalent behaviour, one can be sure to have no common behaviour within Q_A and Q_B. On the other hand, if there is a pair of

equivalent islands, one continues by checking their direct predecessors in both matrices Q_A and Q_B, and if there are equivalent ones one has to compare also the transition matrices from there to the absorbing islands. This procedure has to be repeated until no more equivalent islands are found.

After having determined the equivalent parts, both matrices Q_A and Q_B are reordered as shown in Fig. 7 where the matrix Q_C contains the common behaviour.

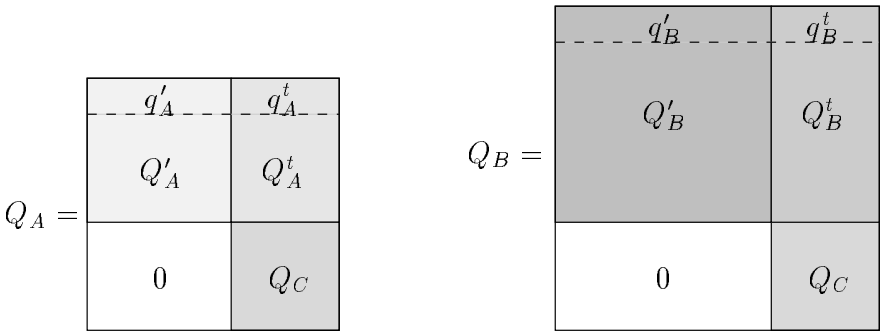

Figure 7: Matrices Q_A and Q_B before the application of the choice operator. Islands and states within islands are reordered such that the common behaviour is described by the common submatrix Q_C.

In general, the submatrices Q'_A, Q'_B and Q_C are not single islands but comprise groups of islands. For the computation of the matrix Q_{A+B} the now reordered matrices Q_A and Q_B have to be composed in the way shown in Fig. 8. As mentioned above, it has to be checked whether the old initial states of A and B are still reachable. If not, these states have to be eliminated.

Further remarks on the normal form We have introduced the normal form of a transition rate matrix as our standard way for describing systems. It is worth mentioning that this form is preserved by all other operators which we have discussed up to now, namely 0, X and prefixing. This observation is trivial for 0 and X because the corresponding matrices contain just one element. The normal form of the matrix $Q_{(a,\lambda).A}$ follows obviously if a new state is added to the matrix Q_A which is already in normal form. Even in the case when a permutation is necessary (as described in Sec. 3.1) this remains true: State i, the state which is equivalent to the new initial state, must belong to the first island, because the first state is reachable from state i. Thus the permutation only affects the first island, but does not destroy the required normal form.

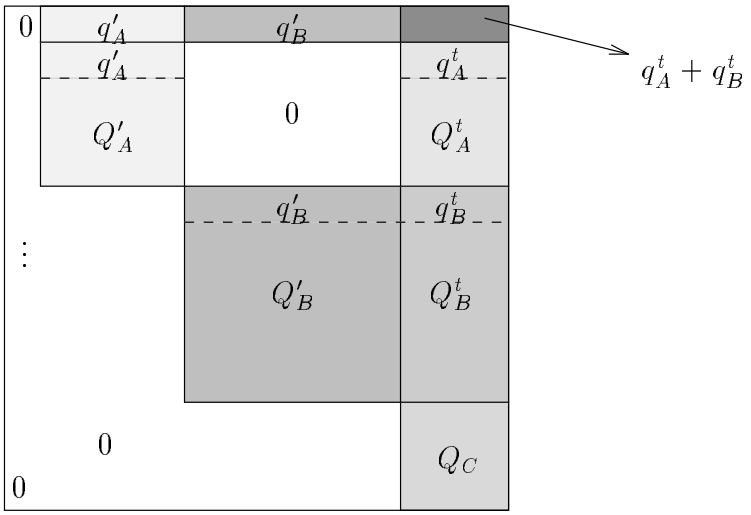

Figure 8: Combination of the reordered matrices Q_A and Q_B for implementing the choice operator

3.3 Recursion

In this subsection we describe the matrix semantics for the recursion operator, i.e. we are interested in the matrix associated with the process term $recX : A$. In the trivial case where A does *not* contain the process variable X the recursion operator takes no effect, such that $[\![recX : A]\!]_{MS} = [\![A]\!]_{MS}$. In the interesting case the process term A has one or more occurrences of X which implies that $Q_A = [\![A]\!]_{MS}$ is of the form shown in Fig. 9 (left). There is one absorbing island containing the single state X. It should be noted that even if A contains more than one X the matrix Q_A will still have only one entry X, assuming that the matrix Q_A is of RMI-type. Several X's in Q_A would have been detected as common behaviour in an earlier step and replaced by a single island X.

The basic idea for the semantics of the recursion operator is simple. For the moment, let the X-island be placed at the last position of the matrix. Entering state X has the same effect as entering the initial state of the process term. Therefore, as depicted in Fig. 9, column q^X_A of the matrix Q_A is added to the first column, and afterwards the row and column corresponding to X are eliminated.

We observe that the matrix resulting from this simple scheme is not necessarily in normal form, the reason being that we have assumed that the X-island is in the last position. For an explanation, see the Fig. 10, which shows two possible normal forms of an example matrix Q_A (top). If the recursion semantics is applied to the matrix on the left, the resulting matrix will not be in normal form. On the other hand, applying recursion semantics to the matrix on the right results in a matrix of normal form. This matrix has just two islands: The new initial island corresponding to the former islands

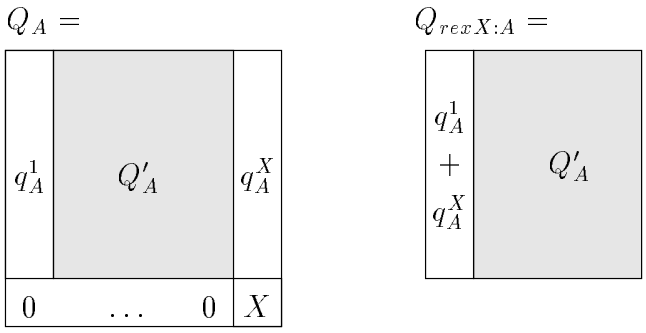

Figure 9: Basic idea for recursion

Q_1 and Q_3, and the remaining old island Q_2. In both cases, submatrix Q'_1 is obtained from submatrix Q_1 by adding to its first column the vector q_1^X. We observe that the resulting matrix will be in normal form if and only if the following condition holds: All islands from which X cannot be reached must be positioned after X.

In addition, there is the problem that after following the basic idea for recursion the resulting matrix may violate condition M. It is clear that the only part of the matrix which can possibly contain equivalent states is the new initial island which is formed by a combination of all the original islands from which X was reachable. The new initial island differs from the corresponding part of the old matrix only in the first column (to which the column X was added). For an example of this situation cf. Fig. 11.

The general treatment of all such possible situations seems to be very hard. However, it will not exceed the effort for general lumping within the first new island. There is furthermore a strong guess that one has to consider only *pairs* of states, where one state is out of the old initial island and the other one is new. Further conditions might be found such that the effort for lumping of the new initial state can be decreased.

3.4 Replication

The new replication operator produces a state space which *always* includes subsets of equivalent states. It is an important property that, due to the regularity of the state space, these subsets are easily identified, i.e. it is not difficult to generate a minimal matrix $[\![\, !_S^n A]\!]_{MS}$. The considerations in this subsection are based on the general work on symmetries resulting from the replication of subsystems as described in [Sie94].

Let us for the moment assume that the set of synchronising actions S is empty. It is well known that, given the matrix Q_A, the matrix describing the behaviour of $!_\emptyset^n A$ is given by the tensor expression

$$Q = [\![\, !_\emptyset^n A]\!]_{MS} = \underbrace{Q_A \oplus Q_A \oplus \ldots \oplus Q_A}_{n \text{ times}}$$

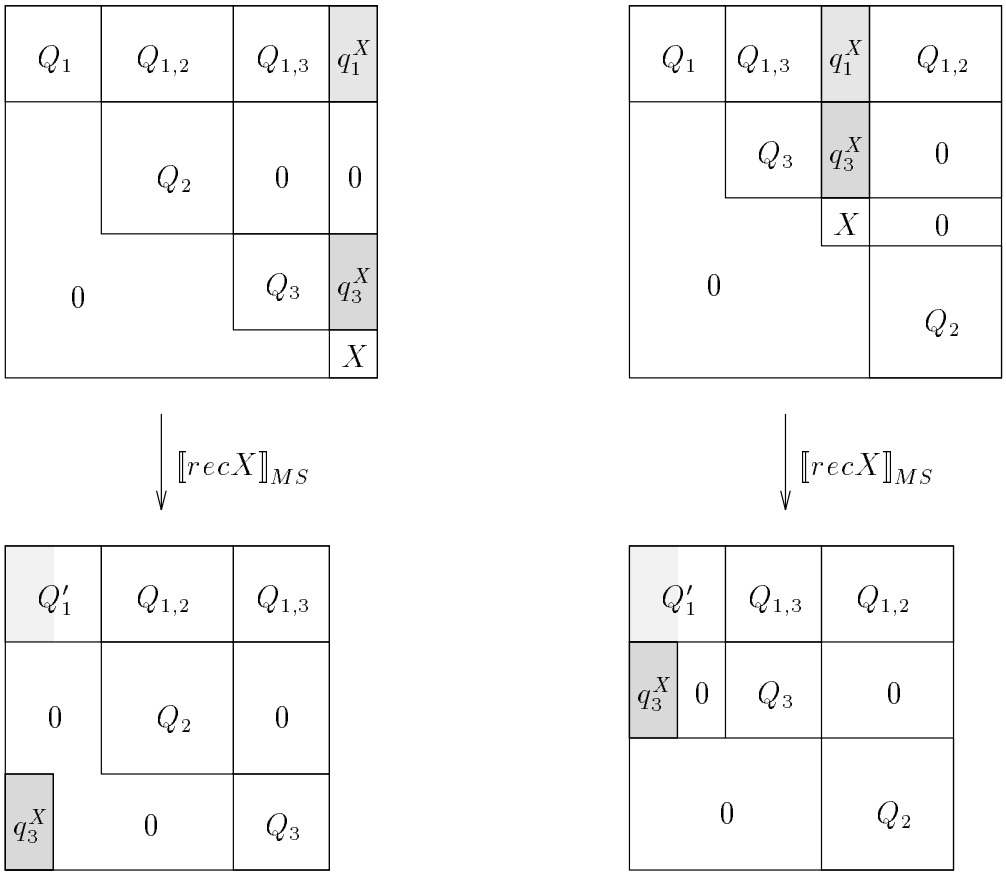

Figure 10: Recursion: Different forms of $Q_A = [\![A]\!]_{MS}$. Only on the right is the resulting matrix of normal form.

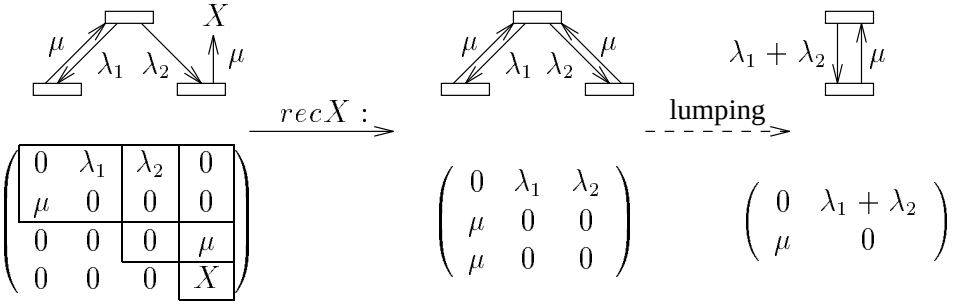

Figure 11: Recursion example that makes further lumping necessary

For an introduction to tensor algebra refer to [Dav81].

Assuming that Q_A is of dimension s, it follows that Q is of dimension s^n. The s^n states of Q can be numbered using n-tuples built from the digits $\{0, 1, \ldots, s-1\}$ in

ascending lexicographical ordering:

$$\begin{array}{rcl} \text{state } 0 & = & (\ 0\ ,\ 0\ ,\ \ldots\ ,\ 0\ ) \\ \text{state } 1 & = & (\ 0\ ,\ 0\ ,\ \ldots\ ,\ 1\ ) \\ & \vdots & \\ \text{state } s^n-1 & = & (\ s-1\ ,\ s-1\ ,\ \ldots\ ,\ s-1\ ) \end{array}$$

In general, the tuple $(i_1, i_2, \ldots, i_n)$ denotes a state in which the k-th replica of the process A is in state i_k, $k = 1 \ldots n$. From symmetry considerations we know that all states marked by tuples which are just permutations of each other, constitute a subset of equivalent states. In total, there exist $\binom{n+s-1}{s-1}$ such different subsets of equivalent states.

The minimal matrix $[\![\ !^n_{\emptyset} A]\!]_{MS}$ describing the behaviour of $!^n_{\emptyset} A$ therefore has $\binom{n+s-1}{s-1}$ states. One can think of two possible ways for its construction from Q_A, see Fig. 12.

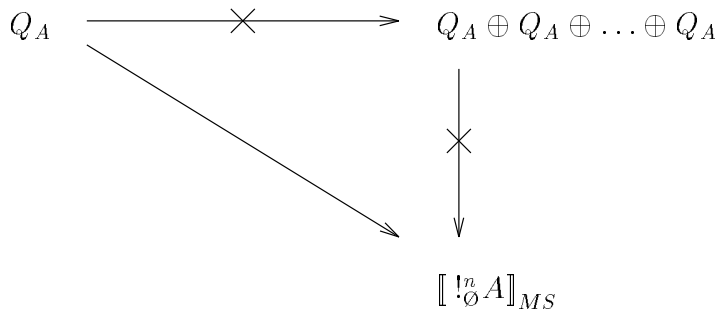

Figure 12: Two ways for constructing $[\![\ !^n_{\emptyset} A]\!]_{MS}$

The explicit computation of the tensor sum and the subsequent combination of equivalent states would be extremely costly. We have seen that the intermediate matrix is of dimension s^n ! Instead we propose an algorithm which generates the reduced matrix $[\![\ !^n_{\emptyset} A]\!]_{MS}$ directly from Q_A, corresponding to the direct arrow in Fig. 12.

Rather than giving a complicated formal description of the algorithm we will explain the scheme by using an example. In Fig. 13 a matrix Q_A of dimension 3 is given. For simplicity its non-zero transition rates are chosen to be different natural numbers. This choice of transition rates makes it easy to follow the example because no ambiguity will arise. The figure also depicts the matrices $[\![\ !^2_{\emptyset} A]\!]_{MS}$ and $[\![\ !^3_{\emptyset} A]\!]_{MS}$. It can be observed that the entries of the latter are derived in a straight-forward manner from the entries of Q_A.

For formally specifying the entries of the matrix $Q = [\![\ !^n_{\emptyset} A]\!]_{MS}$ we define the help function $\mathrm{ndiff}(t_1, t_2)$ which returns the number of digits which are different between the tuples t_1 and t_2, where the position of a digit within the tuple is irrelevant (i.e. tuples are regarded as multi-sets). For example, $\mathrm{ndiff}(000, 001) = 1$, $\mathrm{ndiff}(011, 112) = 1$ and

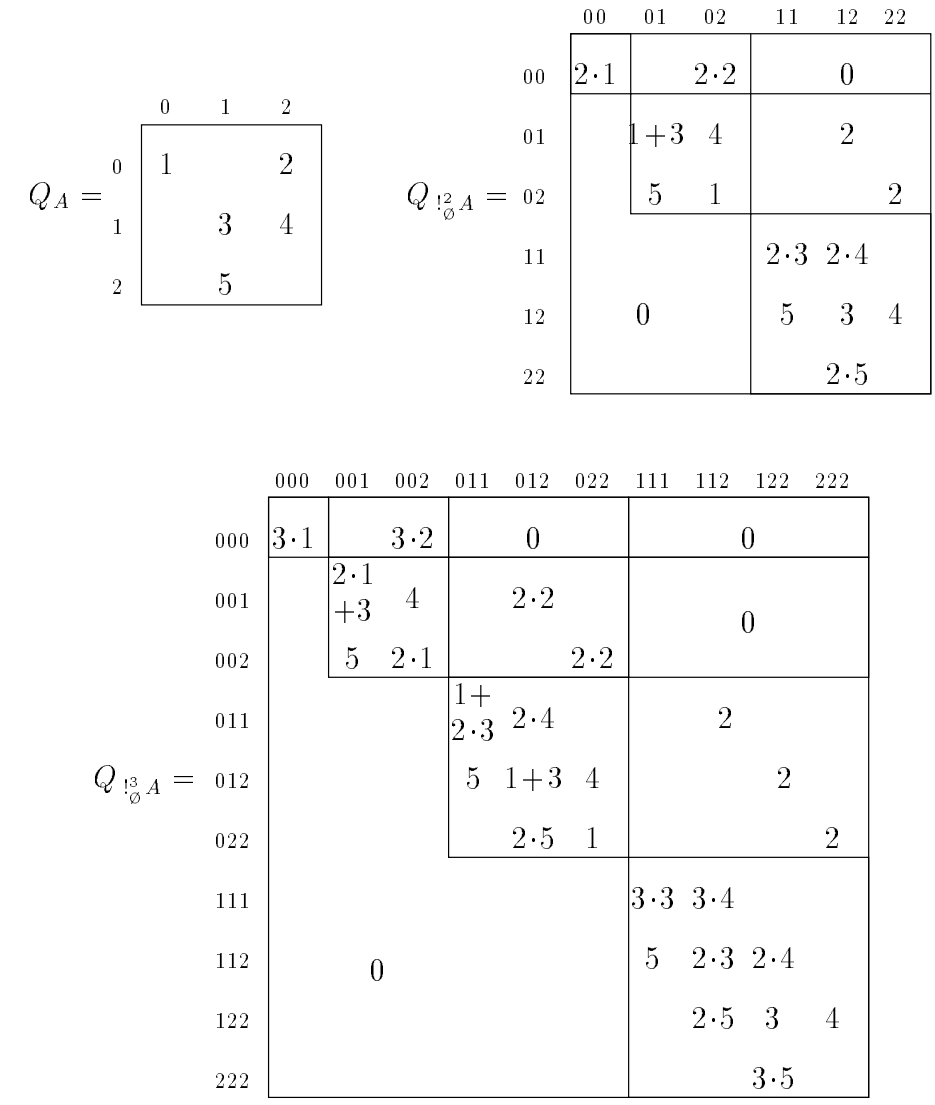

Figure 13: Example for constructing $[\![\, !^n_\emptyset A]\!]_{MS}$

$\text{ndiff}(011, 122) = 2$. We further define the weight $w_i(t)$ of a digit in a tuple as the number of occurrences of the digit i in the tuple t. For instance, $w_0(001) = 2$ and $w_1(001) = 1$.

Let o and t be the tuples denoting the originating state and the target state of a transition, respectively. Then the transition rate from the originating to the target state is of the general form

$$r_{o,t} = \begin{cases} \sum_{i \in o} w_i(o) a_{ii} & \text{if } \text{ndiff}(o,t) = 0 \\ w_i(o) a_{ij} & \text{if } \text{ndiff}(o,t) = 1 \\ 0 & \text{otherwise} \end{cases}$$

The first of the three cases corresponds to the entries on the diagonal of Q. If there is a self-loop from state i to state i in Q_A, its rate a_{ii} is multiplied by the weight of the digit i.

In the second case, the target state differs from the originating state by exactly one digit (again disregarding positions). We assume that a digit has changed from i to j. Then the transition rate is given by a_{ij} multiplied by the weight of the digit i in the originating state.

In the third case, the target state differs from the originating state in more than one digit. Since simultaneous transitions occur with probability 0 in a Markovian environment, the transition rate is 0 in this case.

Before we conclude this subsection, two important questions must be addressed:

- In the case where $S \neq \emptyset$ the matrix Q can be computed in a similar way. Assuming $S = \{a\}$ we have

$$r_{o,t} = \begin{cases} \sum_{i \in o} a_{ii} & \text{if } t = o \\ a_{ij} & \text{if } t = o_{i:=j} \\ 0 & \text{otherwise} \end{cases}$$

  Here, $o_{i:=j}$ denotes the tuple o with all entries i changed to j.

- If Q_A is reducible, i.e. if it consists of more than one island, the states of the matrix Q must be generated according to the island structure of Q_A in order to keep the resulting matrix Q in normal form.

4 Future Work and Conclusion

The language TIPP^{MS} has several obvious shortcomings which will be addressed briefly now:

- It is necessary to enhance the set of actions Act such that more than one action type is permitted. This can be dealt with in a straight forward way: Matrix entries become tuples, one component for each action $a \in Act$. Checking for equivalence does then mean to check the behaviour with respect to *all* action types. Two states can be considered equivalent only if equivalent behaviour is ensured for *all* action types.

- The language TIPP^{MS} lacks a general parallel composition operator $A \parallel_S B$. Only the case $A \neq B$ is of interest here, otherwise we can refer back to the new replication operator. Given the two matrices $Q_A = Q_{A,l} + Q_{A,e}$ and $Q_B = Q_{B,l} + Q_{B,e}$ (the transitions corresponding to the action e are extracted into a special matrix), we have the known relations

$$\begin{aligned} Q_{A \parallel_\emptyset B} &= Q_A \oplus Q_B \\ Q_{A \parallel_{\{e\}} B} &= Q_{A,l} \oplus Q_{B,l} + Q_{A,e} \otimes Q_{B,e} \end{aligned}$$

but some problems regarding minimality and reachability have to be studied. If A and B have common behaviour, the resulting matrix is lumpable in a fashion related to the lumping of processes generated by replication. If $S \neq \emptyset$ there is a potential problem with unreachable states which have to be eliminated in an extra step.

- We intend to enhance TIPP^{MS} by a hiding operator.

The proposed way of generating a transition rate matrix from a process term has some significant advantages over traditional approaches. Applying the concept of lumpability to an existing very large state space requires a lot of memory, often more than what is available, thus making the approach infeasible. In contrast, applying our proposed method for the generation of the state space leads to a minimal transition rate matrix in every single step of the construction procedure. Therefore there is never a waste of memory space.

When analysing large and complex process terms the matrix generation can be easily parallelised according to the structure of the given process term. Sub-expressions can be translated into their corresponding matrices independently of each other. Only the last step (the composition at the highest level) must be performed in a completely sequential manner.

Besides the fact that the resulting matrix is minimal, i.e. not further lumpable, this matrix is also in normal form. Because of the special block diagonal structure of this matrix smart algorithms for stationary or transient analysis can be applied. As a simple example, applying stationary analysis to a normal form matrix with a single absorbing island results in the same state probabilities as applying stationary analysis to the absorbing island directly.

References

[Buc94a] P. Buchholz. Exact and Ordinary Lumpability in Finite Markov Chains. *Journal of Applied Probability*, 1994. scheduled March 1994.

[Buc94b] P. Buchholz. On a Markovian Process Algebra. Forschungsbericht 500, Informatik IV, Universität Dortmund, 1994.

[Dav81] M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, February 1981.

[GHR93] Norbert Götz, Ulrich Herzog, and Michael Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *Proc. of the 16th Int'l Symposium on Computer Performance Modelling, Measurement and Evaluation, PERFORMANCE '93*. Springer, 1993. LNCS 729.

[Göt94] Norbert Götz. *Stochastische Prozeßalgebren – Integration von funktionalem Entwurf und Leistungsbewertung Verteilter Systeme.* Dissertation, Universität Erlangen, 1994.

[Hil93] J. Hillston. PEPA - Performance Enhanced Process Algebra. Technical Report CSR-24-93, Dept. of Computer Science, University of Edinburgh, March 1993.

[Hil94] J. Hillston. *A Compositional Approach to Performance Modelling.* PhD thesis, Department of Computer Science, University of Edinburgh, 1994.

[HR94] H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proceeding of the 2nd Workshop on Process Algebra and Performance Modelling.* University of Erlangen-Nürnberg, IMMD, 1994.

[KS76] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Springer, 1976.

[Sie94] M. Siegle. Reduced Markov Models of Parallel Programs with Replicated Processes. In *2nd EUROMICRO Workshop on "Parallel and Distributed Processing"*, pages 126–133, Malaga, Spain, January 1994.