Compositional Reduction of Performability Models based on Stochastic Process Algebras

Ulrich Klehmet and Markus Siegle
University of Erlangen-Nürnberg, IMMD 7, Martensstr. 3,
91058 Erlangen, Germany
email: klehmet@informatik.uni-erlangen.de

KEYWORDS

Stochastic process algebra, bisimulation equivalence, Markov reward model, compositional reduction

ABSTRACT

Stochastic Process Algebras (SPA) have been proposed as $compositional$ specification formalisms for quantitative models. Here we apply these compositional features to SPAs extended by rewards. State space reduction of performability models can be achieved based on the behaviour-preserving notion of Markov Reward Bisimulation. For a framework extended by immediate actions we develop a new equivalence relation which allows further model reduction. We show that both bisimulations are *congruences* concerning the composition operators of the SPA, which enables a *compositional reduction technique*.

INTRODUCTION

Classical process algebras (e.g. CCS (Milner 1989), CSP (Hoare 1985), LOTOS (Bolognesi and Brinksma 1987)) were designed as formal description techniques for concurrent systems. Basically, a process algebra simply provides a language for describing systems as a cooperation of smaller components, which themselves belong to the language. The basic constructs for all specifications are $actions$ and $processes$, where processes may perform actions. The Stochastic Process Algebra (SPA) modelling paradigm is aimed at the integration of functional and temporal aspects in a single specification and modelling approach for distributed systems (Götz *et al.* 1993), (Hillston 1996) (e.g. multiprocessor systems, communication networks, production lines, workflow systems, …). In order to achieve this integration, temporal information is attached to actions, in the form of continuous random variables representing activity durations. In addition, we integrate rewards into SPAs, reflecting the measure of interest concerning the performance and dependability evaluation. With rewards as part of the syntax of a SPA we bridge the gap between high-level functional specification and low-level definition of quantitative measures, see (Klehmet 1998) for details.
The concept of stochastic process algebras follows the lines of classical process algebras: The system behaviour is described by an abstract language from which a *labelled transition system* (LTS) is generated, using structural operational rules (Hermanns *et al.* 1998b). The additional time and reward information in the semantic model makes it possible to evaluate different system aspects:

- functional behaviour (e.g. liveness or deadlocks)
- temporal behaviour (e.g. throughput, waiting times, reliability)
- combined properties (e.g. probability of timeout, duration of certain event sequences)

A special feature of (S)PAs is *compositionality*, which means that complex models can be constructed in a stepwise fashion out of smaller building blocks. In such a way it is possible to build highly modular and hierarchical system descriptions using the composition operators of the SPA-specification language. Among other operators, a parallel composition operator is used to express concurrent execution and possible synchronisation of processes. An *abstraction* mechanism provides means for treating components as *black boxes*, making their internal behaviour transparent for the environment. One of the distinguishing features of process algebras is the existence of different notions of *equivalence* of processes. These basic features of (S)PAs, namely compositionality, abstraction and equivalence, often summarised by the term $constructivity$, are graphically depicted in Fig. 1.

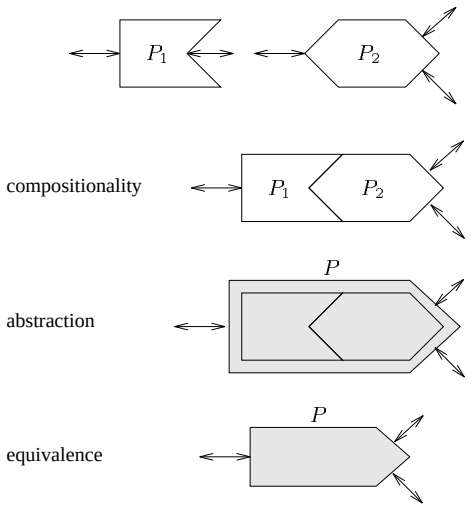

Figure 1: Constructivity – basic principle

Equivalence relations allow the comparison of system descriptions for the purpose of validation or for model reduction (a class of equivalent states can be replaced by a single macro state). Dependent on the point of view and the current aim of analysis, different concepts of equivalence may apply. Formal definition of equivalence supports transformations from one description to another while similar behaviour is guaranteed. The most important class of equivalence relations is the

class of *bisimulation equivalences* or *bisimilarities* (Hermanns *et al.* 1998b). The basic idea is that bisimilar systems can simulate each other's behaviour.

AN EXTENSION OF MARKOVIAN REWARD BISIMULATION

We first consider the language $\mathcal{L}_1$ which is given by the following grammer:

$$P ::= \quad Stop \quad | \quad (a, \lambda, r); P \quad | \quad P[]P \quad | \quad P|[S]|P \quad | \quad hide\ S\ in\ P$$

The productions' meaning is (from left to right) inactivity, prefixing, choice, parallel composition and hiding. The prefix by action a has two additional parameters: The rate λ which is the parameter of the exponentially distributed delay, and the reward rate r. The reward, a real number or a functional expression, reflects the quantitative characteristics or enables the computation of the measures of interest (see (Klehmet 1998), for instance, for a further explanation of these language elements). In (Klehmet 1998) we defined a performance- and dependability-related equivalence relation for $\mathcal{L}_1$, called *Markovian Reward Bisimulation* ($\sim_{MR}$), which is based on Markov chain *lumpability* (Nicola 1990). In addition to functionally and temporally equivalent behaviour, equivalence of processes according to $\sim_{MR}$ guarantees that the same rewards are derived. This means for process terms P, Q with $P \sim_{MR} Q$ we are sure to observe the same properties concerning the quantitative modelling.

Definition 1 *Markovian Reward Bisimulation*
P and Q are Markovian Reward bisimilar, written $P \sim_{MR} Q$, if they are contained in an equivalence relation $\mathcal{S}$ on $\mathcal{L}$ such that each $(\widehat{P}, \widehat{Q}) \in \mathcal{S}$ implies for all $a \in Act$ and all $C \in \mathcal{L}/\mathcal{S}$:

i) $\gamma_M(\widehat{P}, a, C) = \gamma_M(\widehat{Q}, a, C)$ where:

$$\gamma_M(R, a, C) = \sum_{\lambda} \lambda,\ \lambda \in \{|\lambda | R \xrightarrow{a,\lambda,\cdot} R';\ R' \in C|\}$$

ii) $\gamma_R(\widehat{P}, a, C) = \gamma_R(\widehat{Q}, a, C)$ where:

$$\gamma_R(T, a, C) = \sum_{r} r,\ r \in \{|r | T \xrightarrow{a,\cdot,r} T';\ T' \in C|\}$$

(we use $\{|\cdots|\}$ to denote multiset brackets)

With Markovian Reward Bisimulation we have the possibility of reducing the state space of the underlying Markov Chain (MC) and therefore the Markov Reward Model (MRM). However, an important question arises: Is there another equivalence relation which guarantees the same reward and results, but enables further state space reduction? Using the hiding operator we can abstract from actions which are not relevant to a specific performance or dependability analysis. But unfortunately, relaxing the observability of actions by making them invisible (through the $hide$-operator any named action a is changed to the special invisible action τ) is not sufficient in the stochastic setting, because *only the name of the action is invisible*. The stochastic delay of an action (τ, λ, r) remains visible for an observer. Consider a process $(a, \mu, r_\mu); (\tau, \lambda, r_\lambda); Stop$. As the sequence of two exponentially distributed phases is no longer exponentially distributed, it is impossible to find a rate ν such that $(a, \nu, r); Stop$ is equivalent to the above process with respect to their random distributions. But there is another observation: Some activities may require only a very small amount of time compared to the duration of other activities. They can be considered as non-time-consuming at the level of detail at which the model is developed. Therefore, only their functional (and not their temporal and performance-related) behaviour is significant to modelling. For instance, a model of a multiprocessor system described at a high level of abstraction may neglect the durations of task switching, since these operations require a very small amount of time in contrast to task executions. We call such events *immediate* actions, denoted by $\underline{a}$, $\underline{b}$, ... and add them to the language $\mathcal{L}_1$. This gives rise to the language $\mathcal{L}_2$ whose grammar is as above but extended by the production

$$P \quad ::= \quad \underline{a}; P$$

The corresponding set of action names of $\mathcal{L}_2$ is now
$Act_{\mathcal{L}_2} = Act_M \cup \{\tau\} \cup Act_I$ with
– Act_M the set of (Markovian) delayed actions a, b, ...
– τ the (immediate) invisible action and
– Act_I the set of (visible) immediate actions $\underline{a}$, $\underline{b}$, ...
We allow synchronization of immediate actions as well as Markovian actions but not between Markovian and immediate actions.

Example:
As an example we consider a simple queueing system. It consist of an arrival process $Arrival$, a queue with finite capacity named $Queue$ and a $Server$. We assume that the process of incoming arrivals – modelled by action $arrive$ – is a Poisson process with rate λ. It is followed by an enqueue action $\underline{enq}$ which has negligible time delay. Therefore it is specified as an immediate action.

$$Arrival := (arrive, \lambda, 0);\ \underline{enq};\ Arrival$$

The behaviour of a finite queue can be described by a family of processes, one for each value of the current queue population. Depending on the population, the queue may permit to enqueue a job ($\underline{enq}$), dequeue a job ($\underline{deq}$) or both. The latter possibility is described by a *choice* operator [] between two alternatives. Again, dequeueing happens without relevant time delay, i.e. it is modelled as an immediate action, too.
$Queue_0 := \underline{enq};\ Queue_1$
$Queue_i := \underline{enq};\ Queue_{i+1}\ []\ \underline{deq};\ Queue_{i-1} (1 \leq i < max)$
$Queue_{max} := \underline{deq};\ Queue_{max-1}$
Next, we define a server process whose service time is exponentially distributed with rate μ.

$$Server := \underline{deq};\ (serve, \mu, 1);\ Server$$

These separate processes can now be combined by the *parallel composition* operator $|[\ldots]|$ in order to describe the whole queueing system. This operator is parametrised with a list '...' of actions on which the partners are required to synchronise:

$$System := Arrival \quad |[\underline{enq}]| \quad Queue_0 \quad |[\underline{deq}]| \quad Server$$

Considering the rewards, we want to compute the $utilisation$ of the server. The server of the queueing system will be utilised whenever it performs a $serve$ activity. Therefore the reward function results in a mapping of the Markovian actions of the process $System$ as follows:

$$reward : \{Act_M\}_{System} \longrightarrow \{0, 1\} \quad \text{with}$$
$$reward(serve) := 1$$
$$reward(arrive) := 0$$

A formal semantics of our language $\mathcal{L}_2$ associates each language expression with an unambiguous interpretation, a *labelled transition system* (LTS). It is obtained by structural operational rules which define for each language expression a

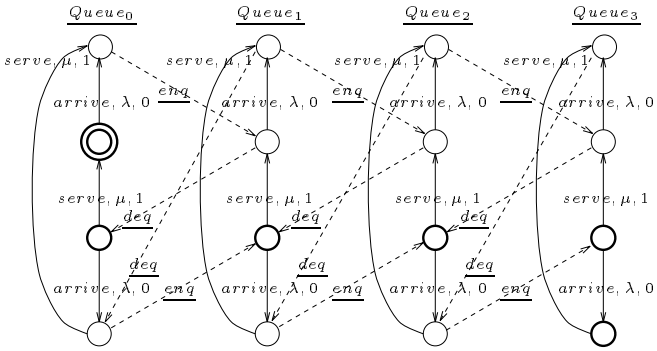

Figure 2: Semantic model of the example (LTS)

specific LTS as the unique semantic model. Fig. 2 shows the semantic model for our example queueing system (assuming that the maximal population of the queue is $max = 3$). There are 16 states, the initial state being indicated by a double circle. Note that there are two kinds of transitions between states: Timed transitions (representing Markovian delayed actions) which are associated with an exponential delay and a reward, and immediate transitions which happen as soon as the respective action is enabled. The timed transitions are drawn by solid arrows. They are labelled with an action name, a rate and the corresponding reward ($\xrightarrow{action,rate,reward}$). The immediate transitions, represented by dashed arrows, are labelled with the name of the corresponding immediate action ($\dashrightarrow^{action}$).

As we can see in Fig. 2, immediate actions imply a semantic model that cannot be directly transformed into a CTMC or MRM, because it may contain instantaneous transitions between states, which are not allowed in a CTMC. However, both visible immediate actions $\underline{a}$, $\underline{b}$, ... and invisible immediate actions τ do not have any impact on the quantitative properties. Because of this fact and based on the semantic model we extend the $\sim_{MR}$-equivalence and define *Weak Markovian Reward Bisimulation* ($\sim_{WMR}$). It is obtained from the Markovian Reward Bisimulation by replacing $\dashrightarrow^{\underline{a}}$ with $\Longrightarrow^{\underline{a}}$. Here $\Longrightarrow^{\underline{a}}$ denotes an (observable) immediate action that is preceded and followed by an arbitrary number (including zero) of invisible immediate actions, i.e. $\Longrightarrow^{\underline{a}} := \dashrightarrow^{\tau^*}\dashrightarrow^{\underline{a}}\dashrightarrow^{\tau^*}$. If $\underline{a}$ is internal ($\underline{a} = \tau$), $\Longrightarrow^{\underline{a}}$ abbreviates $\dashrightarrow^{\tau^*}$.
The main properties of $\sim_{MR}$ were

$$\gamma_M(\widehat{P}, a, C) = \gamma_M(\widehat{Q}, a, C) \quad \text{and}$$
$$\gamma_R(\widehat{P}, a, C) = \gamma_R(\widehat{Q}, a, C)$$

if $\widehat{P}$ and $\widehat{Q}$ are Markovian Reward bisimilar. But for the extension from $\sim_{MR}$ to $\sim_{WMR}$ we do not have to check this condition for $every$ state of the LTS, but only for those states that cannot immediately and invisibly evolve to another state, i.e. do not posses an emanating immediate τ-transition. The reason for this is a stochastic one: The probability that a continuously distributed duration finishes immediately (i.e. at time zero) is zero whereas the probability for the internal immediate transition to take place immediately is obviously one, since nothing can prevent the internal immediate action. Thus, we do not have to compare distributions for processes that possess an emanating immediate τ-transition. We use $P \not\dashrightarrow^{\tau}$ to denote the absence of such immediate internal transitions.

Definition 2 *Weak Markovian Reward Bisimulation*
P and Q are Weak Markovian Reward bisimilar, written $P \sim_{WMR} Q$, if they are contained in an equivalence relation $\mathcal{S}$ on $\mathcal{L}_2$ such that each $(\widehat{P}, \widehat{Q}) \in \mathcal{S}$ implies for all $\underline{a}$, $a \in Act_{\mathcal{L}_2}$ and all equivalence classes $C \in \mathcal{L}_{2/\mathcal{S}}$:

- if $\widehat{P} \Longrightarrow^{\underline{a}} \widehat{P'}$ then there is $\widehat{Q'}$ such that $\widehat{Q} \Longrightarrow^{\underline{a}} \widehat{Q'}$ with $(\widehat{P'}, \widehat{Q'}) \in \mathcal{S}$
- if $\widehat{P} \Longrightarrow^{\tau} \widehat{P'} \not\dashrightarrow^{\tau}$ then there is $\widehat{Q'}$ such that $\widehat{Q} \Longrightarrow^{\tau} \widehat{Q'} \not\dashrightarrow^{\tau}$ with $(\widehat{P'}, \widehat{Q'}) \in \mathcal{S}$ and
  $-\ \gamma_M(\widehat{P'}, a, C) = \gamma_M(\widehat{Q'}, a, C)$
  $-\ \gamma_R(\widehat{P'}, a, C) = \gamma_R(\widehat{Q'}, a, C)$

with the meaning of expressions $\gamma_M(\,)$ and $\gamma_R(\,)$ like in Definition 1.

Two processes P and Q are Weak Markovian Reward bisimulation equivalent, written $P \sim_{WMR} Q$, if $(P, Q) \in \mathcal{B}$ for some Weak Markov Reward Bisimulation $\mathcal{B}$. We see, the relation $\sim_{WMR}$ extends $\sim_{MR}$ by additionally allowing abstraction from *internal immediate* actions. Based on this equivalence relation we can obtain reduced MRMs, preserving their performance/dependability features, but with a smaller state space.

Let us return to the previous example. We know that from the point of view of performance/dependability modelling, e.g. the utilisation of the server, the immediate actions $\underline{enq}$ and $\underline{deq}$ do not have any relevance. That means, we are not interested in the internal details of interaction between $Arrival$ and $Queue$, but more in activities described by actions $arrive$ and $serve$. This requires $abstraction$ from internal details, and is achieved by employing the $hide$-operator:

$$\text{hide } \underline{enq}, \underline{deq} \text{ in } System$$

The resulting semantic model is shown in Fig. 3. Here, the pre-

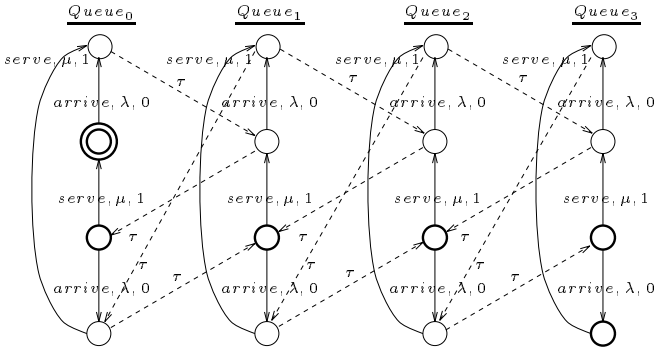

Figure 3: Semantic model after hiding $\underline{enq}$ and $\underline{deq}$

vious actions $\underline{enq}$ and $\underline{deq}$ are changed to the special internal τ-action which is not visible from the environment. Using now the relation $\sim_{WMR}$ defined above, the τ-actions can be eliminated from the semantic model. After applying $\sim_{WMR}$ the resulting semantic model is the LTS in Fig. 4, from which we get the CTMC. After solving the CTMC and taking into consideration the reward function defined before, we can derive the utilisation U of the server as the total reward:

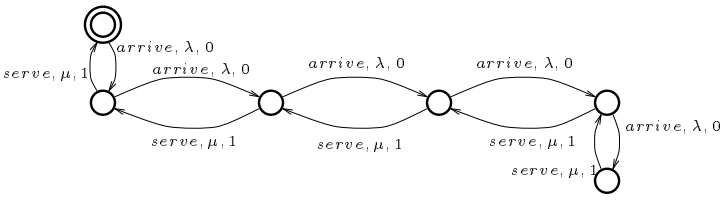

Figure 4: Reduced LTS

$$U := \sum_{P \in \mathcal{L}_2} probability\{P\} * reward(a)$$

$$\text{with} \quad P \xrightarrow{a,\lambda,reward(a)} P'.$$

COMPOSITIONAL REDUCTION

Process algebras are used as abstract concurrent languages. Due to their algebraic structure, process terms reflect compositionality, i.e. how complex terms are composed by operators (such as choice and parallel composition) from simpler ones. In the presence of composition operators it is highly desirable that equivalence notions are *substitutive*. Intuitively, substitutivity allows to replace components by equivalent ones within a larger specification or context $\mathcal{C}$ without changing the overall behaviour. Formally, for an arbitrary context $\mathcal{C}$ which contains subexpression P, written $\mathcal{C} := \mathcal{C}(P)$, and $P \sim Q$, substitutivity implies $\mathcal{C}(P) \sim \mathcal{C}(Q)$. For instance, for process terms P, Q with $P \sim_{MR} Q$:

$$(a, \lambda, r).P \ \sim_{MR} \ (a, \lambda, r).Q \qquad \text{and}$$
$$P|[S]|R \ \sim_{MR} \ Q|[S]|R$$

Substitutive equivalences are also called *congruences* (Hermanns *et al.* 1998a). Practically important for performance and dependability modelling, such equivalences allow *compositional reduction* techniques, where the size of a component's state space may be reduced by the corresponding equivalence, without affecting any significant property of the overall model. This can be exploited in a stepwise reduction of process components and their subsequent composition using the composition operators. Thus, for quantitative modelling it may be possible to reduce the state space of a whole system by dealing only with parts of this system. Compositional reduction has successfully been applied to a variety of systems, see e.g. (Chehaibar *et al.* 1996) for an impressive industrial case study.

We now investigate whether the relations $\sim_{MR}$ and $\sim_{WMR}$ are congruences with respect to the operators of $\mathcal{L}_2$. $\sim_{MR}$ has been proven to be a congruence concerning all operators (Klehmet 1999). On the other hand $\sim_{WMR}$ is a congruence, too, but with the *exception* of the choice operator. In order to illustrate this, we consider the following counterexample. By Definition 2 of $\sim_{WMR}$ it is obvious that

$$\tau; (a, \lambda, r_\lambda); Stop \ \sim_{WMR} \ (a, \lambda, r_\lambda); Stop$$

holds. Supposing that $\sim_{WMR}$ is a congruence with respect to choice, we can conclude that

$$\underbrace{\tau; (a, \lambda, r_\lambda); Stop \ [] \ (b, \mu, r_\mu); Stop}_{P} \quad \sim_{WMR}$$

$$\underbrace{(a, \lambda, r_\lambda); Stop \ [] \ (b, \mu, r_\mu); Stop}_{Q}$$

must also hold. In P the transition labelled (b, μ, r_μ) is not considered in Definition 2 because for P it is not true that $P \stackrel{\tau}{\not\longrightarrow}$. That means with probability one the τ-action will take place in P, i.e. the choice within P is decided in favour of the left side. Thus we have

$$P \ \sim_{WMR} \ \tau; (a, \lambda, r_\lambda); Stop \ \sim_{WMR} \ (a, \lambda, r_\lambda); Stop.$$

On the supposition that $P \ \sim_{WMR} \ Q$ we have:

$$\underbrace{(a, \lambda, r_\lambda); Stop \quad \sim_{WMR} \quad (a, \lambda, r_\lambda); Stop \ [] \ (b, \mu, r_\mu); Stop.}_{Q}$$

But obviously these processes are not equivalent. Thus the assumed congruence property with respect to the choice operator ($[]$) turns out to be false.

We will now demonstrate the performance-conserving compositional reduction through these congruences. Let us return to our queueing system example. We now consider a queueing system with one Poisson arrival process, two queues and two servers. We can build this system from the same components, i.e. processes $Arrival$, $Queue$ and $Server$ are defined as above. The system is now:

$$System := Arrival \ |[\underline{enq}]| \ ((Queue_0 \ |[\underline{deq}]| \ Server) \ |[]| \ (Queue_0 \ |[\underline{deq}]| \ Server))$$

If the queue sizes are given by $max = 3$, the model has 128 states and 384 transitions. By hiding actions $\underline{enq}$ and $\underline{deq}$ and applying Weak Markovian Reward Bisimulation to the complete system, the state space can be reduced to 22 states and 48 transitions.

However, reduction can also be performed in a $compositional$ fashion: The subsystem consisting of one queue-server pair has 8 states and 13 transitions, which can be reduced down to 5 states and 8 transitions. Combining both (reduced) queue-server pairs, we obtain 25 states which can be reduced down to 15 states (this reduction step mainly exploits symmetry of the model). If this reduced system is combined with the arrival process, we get 30 states which can again be reduced to 22 states. This concept of compositional reduction is illustrated in Fig. 5, where the size of the state space and the number of transitions are given for each reduction step. It is interesting

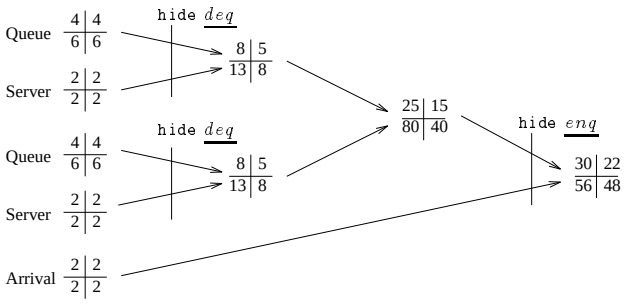

Figure 5: Compositional reduction of the example queueing system

to observe that this system exhibits so-called non-deterministic behaviour: After the completion of a Markovian timed action $arrive$, it is left unspecified which of the two queues synchronises with the arrival process on immediate action $\underline{enq}$ (provided, of course, neither queue is full, in which case the behaviour is deterministic). As a consequence, the Markov chain underlying this specification is formally not completely specified. One may assume that both alternatives occur with the same probability. Alternatively, one may explicitly add infor-

mation (such as a scheduling strategy) in order to resolve nondeterminism.

The equivalences $\sim_{MR}$ and $\sim_{WMR}$ are defined in terms of states and transitions, i.e. on the level of the LTS. The states of the transition system are labelled with the corresponding process descriptions. Therefore, the bisimulation equivalences can be lifted to equalities on the *syntactic* level. These equalities are characterized by a set of *equational laws* and are useful as an additional tool for formal reasoning and as a means of making practical use of equivalences. Some important laws for Weak Markovian Reward Bisimulation are:

$$(a, \lambda, r); \tau; P = (a, \lambda, r); P$$
$$\tau; P \ [] \ (a, \lambda, r); Q = \tau; P$$
$$(a, \lambda, r_1); P \ [] \ (a, \mu, r_2); P = (a, \lambda + \mu, r_1 + r_2); P$$

Such equational laws are the basis for process term rewriting, used for replacing terms by smaller but behaviourally equivalent ones. With the aid of these laws, model reduction can be carried out *before* the LTS is constructed, i.e. purely at the syntactic level.

CONCLUSION

In this paper we showed how one of the main features of Stochastic Process Algebras, namely compositionality, can be applied to performability modelling, i.e. to quantitative modelling. The application of compositionality in the modelling context is two-fold: On the one hand it enables us to build highly modular and hierarchical system descriptions out of smaller components. On the other hand, we use the compositionality property for applying reduction techniques with respect to the state space of Markov Reward Models.

Based on the powerful framework of $bisimulation$ we defined a new equivalence relation, the $Weak\ Markovian\ Reward\ Bisimulation$, which preserves the measures of interest. This equivalence notion is an extension of Markovian Reward Bisimulation defined before. Both equivalence relations are $substitutive$ concerning (almost) all compositional operators of the SPA specification language.
This congruence property allows a stepwise reduction of process components with subsequent composition. That means, it may be possible to reduce the state space of the whole system without the necessity to build the complete state space at any time.

REFERENCES

Bolognesi, T., and Brinksma E. 1987. "Introduction to the ISO Specification Language LOTOS". *Computer Networks and ISDN Systems*, **14**.

Chehaibar, G., Garavel, H., Mounier, L., Tawbi, N., and Zulian, F. 1996. "Specification and Verification of the Powerscale Bus Arbitration Protocol: An Industrial Experiment with LOTOS". *In:* Gotzhein, R., and Bredereke, J. (eds), *Formal Description Techniques IX*. Chapmann Hall.

Götz, N., Herzog, U., and Rettelbach, M. 1993. "Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras". *In: Proc. of PERFORMANCE'93*. Springer LNCS 729.

Hermanns, H., Herzog, U., Klehmet, U., Siegle, M., and Mertsiotakis, V. 1998a. "Compositional Performance Modelling with the TIPPtool". *In:* R. Puignajer, N. Savino, and Serra, B. (eds), *Computer Performance Evaluation*. LNCS 1469. Springer.

Hermanns, H., Herzog, U., and Mertsiotakis, V. 1998b. "Stochastic Process Algebras — Between LOTOS and Markov Chains". *Computer Networks and ISDN Systems*, **30**, 901–924.

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

Hoare, C.A.R. 1985. *"Communicating Sequential Processes"*. Prentice-Hall, Englewood Cliffs, NJ.

Klehmet, U. 1998. ESM'98: "Extensions of Stochastic Process Algebras for Performance and Dependability Modelling". *In:* Zobel, R., and Moeller, D. (eds), *Simulation — Past, Present and Future*.

Klehmet, U. 1999. *"Integration of Markov Reward Models into Stochastic Process Algebras"*. Tech. rept. 2/99. Universität Erlangen-Nürnberg, IMMD 7.

Milner, R. 1989. *"Communication and Concurrency"*. London: Prentice Hall.

Nicola, V.F. 1990. "Lumping in Markov Reward Processes". Research Report RC 14719, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.