Quantifying the Dynamic Behavior of Process Algebras

Peter Buchholz[1]* and Peter Kemper[2]**

[1] Fakultät für Informatik, TU Dresden, D-01062 Dresden, Germany
`p.buchholz@inf.tu-dresden.de`
[2] Informatik IV, Universität Dortmund, D-44221 Dortmund, Germany
`kemper@ls4.cs.uni-dortmund.de`

Abstract. The paper introduces a new approach to define process algebras with quantified transitions. A mathematical model is introduced which allows the definition of various classes of process algebras including the well known models of untimed, probabilistic and stochastic process algebras. For this general mathematical model a bisimulation equivalence is defined and it is shown that the equivalence is a congruence according to the operations of the algebra. By means of some examples it is shown that the proposed approach allows the definition of new classes of process algebras like process algebras over the max/plus or min/plus semirings.

Keywords: process algebras, semiring, bisimulation, congruence.

1 Introduction

Process algebras are one of the most important formal specification techniques to describe the dynamic behavior of discrete event systems. A large variety of different process algebras exists. Process algebras differ in various aspects, e.g., in the way how synchronization is defined, in the used equivalence relations and, may be most important, in the way how dynamics are quantified. First process algebras like Milners CCS [23] or Hoares CSP [21] describe only the functional behavior of systems. Thus, the process algebra specifies which action sequences can potentially occur, but it does not quantify these sequences or relate a notation of time to it. Consequently, process algebras have been extended by relating probabilities to transitions [25,22], by assuming a stochastic timing and assigning rates to transitions [5,8,19,15] and by combining probabilities and time in one model [14]. Also for probabilistic and stochastic process algebras different concrete realizations have been defined which differ in several details. However, almost all process algebras have the following features in common. They have a well defined formal syntax and a semantics defined as some form of a labeled transition system. The specification is compositional. Finally, equivalence

* This research is partially supported by DFG, SFB 358
** This research is partially supported by DFG, SFB 559

relations exist which allow to compare different specifications and certain equivalences are congruences according to the operations of the algebra. In summary, these properties make this model type successful.

Known process algebras describe either labeled transition systems or stochastic processes with transition labels. In the latter case mainly Markov processes or Markov chains are considered. However, recently other algebraic models became very popular for the analysis of discrete event systems, namely min/plus, max/plus or min/max systems [2] with applications in the area of communication protocols [4,10], queueing networks [1] or real time systems [3]. Instead of defining additional specific process algebras for these models, it is more challenging to ask for a common framework which has such process algebras a special instantiations. In this paper we follow the latter approach. We first have to find a general representation of the required operations which can be interpreted adequately in different concrete realizations. It will be shown that the mathematical structure for these operations is a semiring. This structure is very general and contains the mathematical operations available in different process algebras and also in systems like max/plus algebra. Based on the semiring definition a general process algebra can be defined by including the common operations available in different process algebras like prefix, sum and composition. The underlying semantics is a labeled transition system where labels consist of actions and an additional quantification by elements of the semiring. The interesting point is that the mild requirements of a semiring as an algebraic structure are sufficient to define a bisimulation in this framework, which appears natural and which is a congruence according to the operations of the algebra. This bisimulation is valid independently of the semiring one selects for a concrete application.

The outline of the paper is as follows. In the next section we introduce some basic definitions. Afterwards, in Sect. 3 we present the syntax and an operational semantics of Generalized Process Algebra (GPA). Then a bisimulation is defined and it is proved that this bisimulation is a congruence according to the operations of the algebra. In Sect. 5 we present two concrete realizations of GPA and define implicitly process algebras for max/plus and min/plus by means of two examples. The paper ends with the conclusions which outline subjects for future research.

2 Basic Definitions

Before we present our process algebra, the underlying basic structures are defined. We start with semirings which are later used to define quantitative values of labels.

Definition 1. *A semiring* $(\mathbb{K}, \widehat{+}, \widehat{\cdot}, \mathbb{O}, \mathbb{1})$ *is a set* $\mathbb{K}$ *with binary operations* $\widehat{+}$ *and* $\widehat{\cdot}$ *defined on* $\mathbb{K}$ *such that the following axioms are satisfied:*
$\widehat{+}$ *,* $\widehat{\cdot}$ *are associative, and* $\widehat{+}$ *is commutative, right and left distributive laws hold for* $\widehat{+}$ *and* $\widehat{\cdot}$ *,* $\mathbb{O}$ *and* $\mathbb{1}$ *are the additive and multiplicative identities with* $\mathbb{O} \neq \mathbb{1}$*, and for all* $k \in \mathbb{K}$ $k \,\widehat{\cdot}\, \mathbb{O} = \mathbb{O} \,\widehat{\cdot}\, k = \mathbb{O}$ *holds.*

To make the notation simpler we use sometimes $\mathbb{K}$ for the whole semiring.

Some typical examples for semirings are $(\mathbb{B}, \vee, \wedge, 0, 1)$ the Boolean semiring, $(\mathbb{R}, +, \cdot, 0, 1)$ the semiring over the real numbers with the usual addition and multiplication, $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ the min/+ semiring, $(\mathbb{R} \cup \{-\infty\}, \max, +, \infty, 0)$ the max/+ semiring or $(\mathbb{R} \cup \{\infty\} \cup \{-\infty\}, \max, \min, \infty, 0)$ the max/min semiring. In the sequel we use for the addition $\widehat{+}$ and for multiplication $\widehat{\cdot}$ or no operator symbol as usual in most semirings.

Semiring operations can be easily extended to define operations on vectors and matrices. Let $\mathbf{p} \in \mathbb{K}^n$ be a n-dimensional vector and $\mathbf{Q} \in \mathbb{K}^{n,n}$ a $n \times n$ square matrix both including elements from semiring $\mathbb{K}$, then $\mathbf{q} = \mathbf{pQ}$ equals a vector in $\mathbb{K}^n$ which is defined as

$$\mathbf{q}(j) = \widehat{\sum}_{i=0}^{n-1} \mathbf{p}(i)\mathbf{Q}(i,j) \ .$$

where $\widehat{\sum}_{i=0}^{n-1} a_i = a_0 \ \widehat{+} \ \ldots \ \widehat{+} \ a_{n-1}$. Dot product of vectors, matrix addition and multiplication can be defined similarly.

Now we can define transition systems where transitions are labeled with symbols from a finite alphabet and from a semiring. This notion is subsequently used to define the semantics of our process algebra.

Definition 2. *A (finite) multi labeled transition system (MLTS) is a* 5 *tuple* $MLTS = (\mathcal{S}, Act, \mathbb{K}, T, ini)$*, where* $\mathcal{S}$ *is the state space which is countable (finite), Act is a finite set of transition labels,* $\mathbb{K}$ *is a semiring used for the definition of transition costs,* $T : \mathcal{S} \times Act \times \mathcal{S} \to \mathbb{K}$ *is the transition function,* $ini : \mathcal{S} \to \mathbb{K}$ *is the initialization function.*

An $MLTS$ can be interpreted as a graph where each transition is labeled by a symbol to define the operation and a cost to perform the transition. The meaning of costs can be very different depending on the semiring one considers. If the state space is finite, the $MLTS$ corresponds to an automaton with transition costs or weighted automata, a well known model in automata theory. Each $MLTS$ can be characterized by a set of matrices, one for each $a \in Act$ and a vector. Let $\mathbf{Q}_a$ be the matrix for label $a \in Act$, then $\mathbf{Q}_a(x,y) = T(x,a,y)$. $\mathbf{p}^0$ is the initial vector of the system such that $\mathbf{p}^0(x) = ini(x)$. Function ini gives an initial quantification for all states, which depends on the concrete application, e.g. to consider reachability from a certain state $s_i \in \mathcal{S}$ a common definition of ini is $ini(s) = \mathbb{1}$ if $s = s_i$ and $\mathbb{O}$ otherwise. The set of initial states $\mathcal{S}_{ini}$ contains all $s \in \mathcal{S}$ with $ini(s) \neq \mathbb{O}$.

For many, but not for all, applications, the dynamics of the system can be described by vector matrix products. We briefly describe this kind of dynamics here. Usually one can distinguish between generative and reactive models [25]. In the reactive case, the system reacts to the behavior of some environment which determines the label that occurs next. A state of the system is described by a vector with elements from $\mathbb{K}$. $\mathbf{p}^0$ is the initial state vector and if $\mathbf{p}^k$ describes the current state and $a \in Act$ is the label chosen by the environment, then $\mathbf{p}^{k+1} = \mathbf{p}^k\mathbf{Q}_a$ describes the next state. In a generative situation, the system can decide by itself which transition occurs next. In this situation the next state is computed from $\mathbf{p}^{k+1} = \mathbf{p}^k \widehat{\sum}_{a \in Act} \mathbf{Q}_a$. In both cases we may analyze the system according to the state reached after a transition sequence. Thus, let $\omega \in Act^*$, ω_i be the i-th symbol in ω and $|\omega|$ be the length of ω. $\mathbf{p}^\omega$ is the state reached after observing ω starting with $\mathbf{p}^0$. The vector $\mathbf{p}^\omega$ can be computed as

$$\mathbf{p}^\omega = \mathbf{p}^0 \widehat{\prod}_{i=1}^{|\omega|} \mathbf{Q}_{\omega_i} \ . \tag{1}$$

where $\widehat{\prod}_{i=0}^{n-1} a_i = a_0 \,\widehat{\cdot}\, \ldots \,\widehat{\cdot}\, a_{n-1}$. We may go further in analyzing the behavior of a system by analyzing its generation of actions (traces) or its reaction to the input from the environment. Define $c_\omega = \widehat{\sum}_{s \in \mathcal{S}} \mathbf{p}^\omega(s)$ as the cost of transition sequence ω. Observe that costs can have different meanings in each concrete realization.

Examples: If we use $(\mathbb{B}, \vee, \wedge, 0, 1)$ in the above definition and *ini* assigns 1 to the initial state and 0 to all other states, then the resulting model describes the well known labeled transition system. Vector $\mathbf{p}^k$ describes the set of reachable states after k transitions. Thus, $\mathbf{p}^k(x) = 1$ implies that x is reachable after k transitions and in the reactive case, the observed labels have been defined by the environment. For deterministic systems, each matrix row contains one element equal to 1.

In a similar way, probabilistic systems can be defined. For probabilistic systems, $\mathbf{p}^0$ has to define a probability distribution. Depending on the used semantics, matrices $\mathbf{Q}_a$ are stochastic or substochastic matrices. In the reactive case, matrices $\mathbf{Q}_a$ are stochastic, since the environment selects label a. In the generative case, each $\mathbf{Q}_a$ is a substochastic matrix and $\sum_{a \in Act} \mathbf{Q}_a$ is a stochastic matrix.

For stochastic systems describing continuous time Markov chains (CTMCs) with transition labels [8,15,20] the above method of computing the dynamic behavior stepwise is not directly applicable. However, by applying the well known randomization technique [13], the CTMC with transition labels can be transformed into a discrete time Markov chain (DTMC) with transition labels and a Poisson process. For the DTMC with transition labels vectors $\mathbf{p}^k$ can be computed and the Poisson process determines the number of transitions which occur in any finite interval.

Another, less common example is the use of max/+ as semiring and interpret the transition labels as reward. For this model element $\mathbf{p}^k(x)$ describes the maximal reward gained when reaching state x after k steps starting in some state y with reward $\mathbf{p}^0(y)$. In the reactive case, the labels of the path from y to x are given by the environment.

For the min/+ semiring and interpretation of transition labels as costs, we have a very similar interpretation. Now $\mathbf{p}^k(x)$ contains the minimal costs of reaching state x after k transitions.

3 A General Process Algebra with Transitions Costs

We now define a process algebra denoted as General Process Algebra (GPA) which uses the above concepts. First the syntax of the process algebra is defined, afterwards the semantics is introduced. The syntax definition follows the usual concepts used in process algebras [23,21].

Definition 3. *The set $\mathcal{L}$ of terms in Generalized Process Algebra (GPA) over a set of finite transition labels Act including label τ and a semiring $\mathbb{K}$ is defined by $\mathcal{P} ::= \; 0 \mid (a,k).\mathcal{P} \mid \mathcal{P}+\mathcal{P} \mid \mathcal{P}\|_S\mathcal{P} \mid \mathcal{P}\setminus L \mid \mathcal{P}' = \mathcal{P}$ where $a \in Act$, $k \in \mathbb{K}$ and $S, L \subseteq Act \setminus \{\tau\}$.*

To avoid too many parenthesis we define among the operations the following decreasing priorities: Hiding, Prefix, Composition and Summation. We also allow for recursion by defining equations, i.e. we use constants as abbreviations by defining $A = P$ which means that A is an abbreviation for P. In contrast to $=$ we use $A \equiv B$ to denote that A and B are syntactically identical. A term in the algebra is denoted as an agent.

The intuitive meaning of the operators is as follows:

- *0* : describes the termination of an agent which afterwards cannot perform any actions.
- *(a,k).A* : action a is performed and afterwards the agent behaves like A, the cost of performing action a equal k.
- $A + B$: the agent behaves either like A or like B.
- $A \|_S B$: A and B proceed concurrently and independently on all actions which are not in S. All actions from S have to be performed as joint actions by both agents.
- $A \setminus L$: actions from the set L are hidden, i.e., they become τ actions which are no longer usable in joint actions with an environment.
- *B=A* : describes that agent B behaves like agent A.

The semantics of an agent $A \in \mathcal{L}$ is a $MLTS$. As usual in process algebras, we cannot distinguish between an agent and a state. An agent and all its derivatives form the state space of a system. We use the notation $A \xrightarrow{a,k} B$ for $T(A,a,B) = k$ if $k \neq \mathbb{O}$. Semantic rules are given in an operational style of the form

$$\frac{premise_1 \; \ldots \; premise_n}{conclusion} \; \langle name \rangle \; (condition)$$

to be read as:
If *condition* is satisfied, the rule $\langle name \rangle$ can be applied and it can be deduced that *conclusion* holds in case of the assumptions $premise_1 \ldots premise_n$ hold.

In all semantic rules we define, conditions are introduced such that transitions with cost $\mathbb{O}$ are not explicitly generated. However, as usual in labeled transitions systems, if no transition between two states exists, then there is implicitly a transition with cost $\mathbb{O}$. Observe that the meaning of $\mathbb{O}$ in all semirings is such that $\mathbb{O} \hat{\cdot} a = \mathbb{O}$ which means that the contribution of a zero element to a vector matrix product is zero. For the prefix and choice operators we have the following operational semantics.

$$\frac{}{(a,k).A \xrightarrow{a,k} A} \langle pr \rangle \ (k \neq \mathbb{O}) \qquad \frac{A_j \xrightarrow{a,k} A'}{\sum_{i \in I} A_i \xrightarrow{a,k_\Sigma} A'} \langle + \rangle \begin{pmatrix} j \in I, \\ k_\Sigma \neq \mathbb{O} \end{pmatrix}$$

where $k_\Sigma = \widehat{\sum}_{i \in I} T(A_i, a, A')$ and I is some index set. Note that the choice operator can yield the situation of multiple arcs with same labels between agents. The above rule ensures that these arcs result as one whose transition function takes a single, well-defined value of k_Σ. For parallel composition, the following four semantic rules are defined.

$$\frac{A \xrightarrow{a,k} A' \qquad A' \not\equiv A}{A \|_S B \xrightarrow{a,k} A' \|_S B} \langle \|_{S_1} \rangle \begin{pmatrix} a \notin S, \\ k \neq \mathbb{O} \end{pmatrix} \qquad \frac{B \xrightarrow{a,k} B' \qquad B' \not\equiv B}{A \|_S B \xrightarrow{a,k} A \|_S B'} \langle \|_{S_2} \rangle \begin{pmatrix} a \notin S, \\ k \neq \mathbb{O} \end{pmatrix}$$

$$\frac{A \xrightarrow{a,k} A \ \ or \ \ B \xrightarrow{a,l} B}{A \|_S B \xrightarrow{a,k \widehat{+} l} A \|_S B} \langle \|_{S_3} \rangle \begin{pmatrix} a \notin S, \\ k \widehat{+} l \neq \mathbb{O} \end{pmatrix} \qquad \frac{A \xrightarrow{a,k} A' \qquad B \xrightarrow{a,l} B'}{A \|_S B \xrightarrow{a,k \widehat{\cdot} l} A' \|_S B'} \langle \|_{S_4} \rangle \begin{pmatrix} a \in S, \\ k \widehat{\cdot} l \neq \mathbb{O} \end{pmatrix}$$

Rules $\|_{S_1}$, $\|_{S_2}$, and $\|_{S_3}$ describe how both agents proceed independently on actions $a \notin S$, rule $\|_{S_3}$ considers the special case of independent self loops with the same label. The latter is necessary to avoid a reduction of transitions and to retain a well-defined transition function if a parallel composition without synchronisation applies. If only one condition of the premise in $\|_{S_3}$ is satisfied, say $A \xrightarrow{a,k} A'$, then let $l = \mathbb{O}$ for the definition of the conclusion (and vice versa). Actions $a \in S$ are performed as joined actions. The costs of the joint transitions equals the product $\widehat{\cdot}$ of the costs of both transitions (rule $\|_{S_4}$) with respect to the semiring. The definition of costs of synchronized transitions is one of the most discussed questions during the development of stochastic process algebras. Several approaches have been proposed in the literature [18] and there are arguments for and against any of these solutions. We do not claim that the use of $\widehat{\cdot}$ is always the best way to define the costs. However, it is a natural solution from a mathematical viewpoint and we found reasonable interpretations for this choice in all semirings we considered so far.

The semantics of the hiding operator is defined as

$$\frac{A \xrightarrow{a,k} A'}{A \backslash L \xrightarrow{a,k} A' \backslash L} \langle \backslash_1 \rangle \ \ (a \notin L, \ k \neq \mathbb{O}) \text{ and}$$

$$\frac{A \xrightarrow{a_1,k_1} A' \ \ldots \ A \xrightarrow{a_n,k_n} A'}{A \backslash L \xrightarrow{\tau,k_\Sigma} A' \backslash L} \langle \backslash_2 \rangle \ \ (\{a_1, \ldots, a_n\} \subseteq L \cup \{\tau\}, k_\Sigma \neq \mathbb{O} \ ),$$

where $k_\Sigma = \widehat{\sum}_{i=1}^{n} k_i$. Transitions which are not hidden occur as before and hidden transitions between two states appear as a single transition with label τ where costs are accumulated. Note that a transition with label τ can be part of the premises as well.

All operations we have considered so far, allow only the specification of finite behaviors. To define an agent with an infinite behavior one has to define $A = B$ where A occurs in B. E.g., $A = (a,k).A$ is an equation describing an infinite behavior. The semantics of a constant is then given by

$$\frac{A \xrightarrow{a,k} A'}{B \xrightarrow{a,k} A'} \langle = \rangle \ \ (B = A) \ .$$

We sometimes use also variables for processes which can be bound to agents. We use the notation $A\{B/X\}$ when every occurrence of process variable X in A is replaced by agent B. We denote a variable as free if it is not bound to an agent. Following Milner a term of GPA is denoted as an agent if it contains no free variables.

What we have defined in this section is rather typical for a process algebra with an operational semantics. The only difference is that we allow general semirings to define the quantitative values. Concrete applications of this concept are presented in Section 5. The semantics of an agent A is an MLTS where the state space $\mathcal{S}_A$ contains A and all its derivatives reachable by applying the semantic rules. The initialization function *ini* assigns $\mathbb{1}$ to $B \in \mathcal{S}_A$ if $B \equiv A$ and $\mathbb{O}$ otherwise. In this way agent A *generates* an *MLTS*.

Define $\mathcal{D}[A]$ as the set of all successors of agent A, i.e., $\mathcal{D}[A] = \{B \in \mathcal{L} | \exists a \in Act : A \xrightarrow{a,k} B\}$. Similarly $\mathcal{D}_a[A]$ is the set of all successors of A which are reachable by one a labeled transition and $\mathcal{D}^*[A]$ is the transitive and reflexive closure of relation $\mathcal{D}[A]$ and equals $\mathcal{S}_A$.

4 Bisimulation for GPA

Two of the major advantages of process algebras are compositionality and the availability of equivalences which are congruences according to the operations of the algebra. One of the most famous equivalence relations is bisimulation which is well known for untimed systems [24,23] and has been subsequently extended to probabilistic [22] as well as to stochastic systems [7,8,16,19]. In [9] it has been shown that bisimulation can be defined for automata with transition costs where values of costs are elements of a semiring and the behavior of the automaton is defined using vector matrix operations derived from the addition and multiplication of the semiring. Of course, automata with transition costs are identical to finite state MLTS, the semantic model of GPA. We first introduce bisimulation for GPA and prove afterwards the congruence property according to the operations of GPA.

The definition of bisimulation requires some additional notations as a prerequisite. We restrict bisimulations to equivalence relations and assume that the MLTS is of the finite branching type, i.e., the number of transitions with non-zero costs leaving a state is finite. Let $\mathcal{R}$ be an equivalence relation on $\mathcal{L} \times \mathcal{L}$. $\mathcal{CC}_\mathcal{R}$ is the set of equivalence classes of $\mathcal{R}$, $\mathcal{C}_\mathcal{R}$ is an equivalence class of $\mathcal{R}$ and $\mathcal{C}_\mathcal{R}[B]$ is the equivalence class to which agent B belongs.

Definition 4. *An equivalence relation $\mathcal{R}$ on $\mathcal{L} \times \mathcal{L}$ is a bisimulation if and only if for all $(A, B) \in \mathcal{R}$ and all $a \in Act$:*

1. *$ini(A) = ini(B)$;*
2. *if $A \xrightarrow{a,k} C$[1] for some $C \in \mathcal{L}$, then*

$$\widehat{\sum}_{D \in \mathcal{C}_\mathcal{R}[C]} T(A, a, D) = \widehat{\sum}_{D \in \mathcal{C}_\mathcal{R}[C]} T(B, a, D) \ ;$$

[1] Observe that $A \xrightarrow{a,k} C$ implies $k \neq \mathbb{O}$ which follows from our semantic rules.

3. if $B \xrightarrow{a,k} C$ for some $C \in \mathcal{L}$, then

$$\widehat{\sum}_{D\in\mathcal{C}_\mathcal{R}[C]} T(B,a,D) = \widehat{\sum}_{D\in\mathcal{C}_\mathcal{R}[C]} T(A,a,D)\ .$$

This definition corresponds to the usual definition of bisimulation for untimed [23], probabilistic [22] or stochastic [8] systems. Note that in the classical bisimulation [23], one considers existence of specific agents C, C' which need to be bisimilar which is the same as to ask for elements $D \in \mathcal{C}_\mathcal{R}[C]$. Like for other bisimulations the following theorem holds.

Theorem 1. *Let $\mathcal{R}_1$, $\mathcal{R}_2$ be two bisimulations on $\mathcal{L}$, then $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is also a bisimulation.*

Proof. The proof can be found in [9] and follows the same line of argumentation as the proof for the classical theorem in Boolean systems [23]. □

The theorem implies that the largest bisimulation exists as the union of all bisimulations.

Definition 5. *Two agents A and B are bisimilar, denoted as $A \sim B$, if a bisimulation $\mathcal{R}$ exists such that $(A,B) \in \mathcal{R}$.*

The term equivalence implies that equivalent agents behave identically in some sense. This is also the case for a bisimilar agent in our general setting. An example is given in the following theorem where we come back to the notion of transition cost described in Section 2 with $c_\omega = \widehat{\sum}_{s\in S}\mathbf{p}^\omega(s)$ as the cost of a transition sequence ω.

Theorem 2. *Let A and B be agents generating $MLTS_A$ and $MLTS_B$ with $Act = Act_A = Act_B$. If $A \sim B$, then for each $\omega \in Act^*$, $c^A_\omega = c^B_\omega$ where c^A_ω, c^B_ω are the costs for performing ω in A and B, respectively.*

Proof. The proof follows by induction. We first prove that for all $\mathcal{C}_\mathcal{R} \in \mathcal{CC}_\mathcal{R}$, $k \in \mathbb{N}$,

$$\widehat{\sum}_{C\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[A]} \mathbf{p}^0(C) = \widehat{\sum}_{D\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[B]} \mathbf{p}^0(D)$$

holds. The relation is true because $\mathbf{p}^0(A) = ini(A) = ini(B) = \mathbf{p}^0(B)$ holds due to $A \sim B$ and Def. 4 and $\mathbf{p}^0(C) = \mathbb{O}$ for all $C \not\equiv A \vee B$ according to the definition of *ini* for the MLTS semantics of an agent. Now assume that the relation holds for a string ω. We prove the induction to a string $\omega; a$ where $a \in Act$ and $\omega; a$ is the concatenation of ω and a.

$$\begin{aligned}
&\widehat{\sum}_{C\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[A]} \mathbf{p}^{\omega;a}(C) \\
&= \widehat{\sum}_{\mathcal{C}'_\mathcal{R}\in\mathcal{CC}_\mathcal{R}} \widehat{\sum}_{D\in\mathcal{C}'_\mathcal{R}\cap\mathcal{D}^*[A]} \mathbf{p}^\omega(D) \mathbin{\hat{\cdot}} \left(\widehat{\sum}_{C\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[A]} T(D,a,C)\right) \\
&= \widehat{\sum}_{\mathcal{C}'_\mathcal{R}\in\mathcal{CC}_\mathcal{R}} \widehat{\sum}_{D\in\mathcal{C}'_\mathcal{R}\cap\mathcal{D}^*[B]} \mathbf{p}^\omega(D) \mathbin{\hat{\cdot}} \left(\widehat{\sum}_{C\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[B]} T(D,a,C)\right) \\
&= \widehat{\sum}_{C\in\mathcal{C}_\mathcal{R}\cap\mathcal{D}^*[B]} \mathbf{p}^{\omega;a}(C)
\end{aligned}$$

The first equation holds due to the cost assignment by a vector matrix multiplication, see (1). The second equation is based on the requirements for a bisimulation (Def. 4). The third equation follows from (1) like the first one. The identity of the vectors implies the identities of the cost functions which can be seen by the following equations.

$$c_\omega^A = \widehat{\sum}_{\mathcal{C}_\mathcal{R} \in \mathcal{CC}_\mathcal{R}} \widehat{\sum}_{C \in \mathcal{C}_\mathcal{R} \cap \mathcal{D}^*[A]} \mathbf{p}^\omega(C) \,\widehat{\cdot}\, \left(\sum_{D \in \mathcal{D}^*[A]} T(C,a,D) \right) =$$

$$\widehat{\sum}_{\mathcal{C}_\mathcal{R} \in \mathcal{CC}_\mathcal{R}} \widehat{\sum}_{C \in \mathcal{C}_\mathcal{R} \cap \mathcal{D}^*[B]} \mathbf{p}^\omega(C) \,\widehat{\cdot}\, \left(\sum_{D \in \mathcal{D}^*[B]} T(C,a,D) \right) = c_\omega^B$$

□

The theorem introduces some form of trace equivalence for GPA which is observed by bisimilar agents.

An equivalence relation is especially useful, if it is a congruence according to the operations of the algebra. In this case, equivalence assures that equivalent agents behave "equivalently" in all possible environments and equivalent agents can be substituted which allows the interleaving of composition and aggregation [11,8]. The following theorems 3, 4 state that bisimulation is indeed a congruence according to the operations of GPA.

Theorem 3. *Let $A, B, C \in \mathcal{L}$ with $A \sim B$, then the following relations hold*

1. $(a,k).A \sim (a,k).B$ *for all* $a \in Act$ *and* $k \in \mathbb{K} \setminus \{\mathbb{O}\}$,
2. $A + C \sim B + C$,
3. $A \setminus L \sim B \setminus L$ *for all* $L \subseteq Act \setminus \{\tau\}$, *and*
4. $A \|_S C \sim B \|_S C$ *for all* $S \subseteq Act \setminus \{\tau\}$.

The proofs follow from the proofs for untimed process algebras [23] or stochastic process algebras [8]. Note that we have a strong similarity to untimed process algebras if one focuses only on labels $a \in Act$, such that the crucial point for bisimulation is in the treatment of elements of the semiring. A key observation is that the semantics of Prefix, Choice and Hiding preserve existence of transitions and that if multiple transitions between two states are merged into a single one, elements of the semiring are added by $\widehat{+}$. This matches requirements for the bisimulation which adds elements of the semiring by $\widehat{+}$ as well. Due to associativity and commutativity of $\widehat{+}$ in Def. 1 the order in which elements are added does not matter. We omit proofs of prefix, choice and hiding for Theorem 3 since these are lengthy and do not give further insight. We focus on the proof of composition, since this one is the most interesting, especially rule $\|_{S4}$, where elements of the semiring are multiplied. Observe that the distributive laws of the semiring allow to conclude that for finite index sets I, J, and K and a_i, b_j, c_k elements of the semiring for all $i \in I$, $j \in J$, $k \in K$ holds

$$\widehat{\sum}_{i \in I, k \in K} a_i \,\widehat{\cdot}\, c_k = (\widehat{\sum}_{i \in I} a_i) \,\widehat{\cdot}\, (\widehat{\sum}_{k \in K} c_k) = (\widehat{\sum}_{j \in J} b_j) \,\widehat{\cdot}\, (\widehat{\sum}_{k \in K} c_k)$$
$$= \widehat{\sum}_{j \in J, k \in K} b_j \,\widehat{\cdot}\, c_k$$

if $\widehat{\sum}_{i\in I}a_i = \widehat{\sum}_{j\in J}b_j$. Having this in mind, the argumentation on equations for proving rule $||_{S4}$ are easier to follow. The necessity of associativity and distributivity of $\widehat{\cdot}$ over $\widehat{+}$ for existence of congruences is well known, e.g. [18]. The proof follows the same line of argumentation as Milner's proof of strong bisimulation in [23].

Proof. of composition in Theorem 3
Let $\mathcal{R}$ be an equivalence relation such that $(A\|_S C, B\|_S C) \in \mathcal{R}$ if and only if $A \sim B$. We have to prove that $\mathcal{R}$ is a bisimulation. Since $ini(A\|_S C) = ini(B\|_S C) = \mathbb{1}$ and $ini(D) = \mathbb{O}$ for $D \not\equiv (A\|_S C) \vee (B\|_S C)$ by definition of the MLTS generated by $A\|_S C$, resp. $B\|_S C$, the first condition of a bisimulation is satisfied.

Suppose $(A\|_S C, B\|_S C) \in \mathcal{R}$, let $A\|_S C \xrightarrow{a,k} E$ then there are four cases according to the four semantic rules of composition; cases 1-3 consider $a \notin S$:
<u>Case 1:</u> $A \xrightarrow{a,k} A'$, $A \not\equiv A'$ and $E \equiv A'\|_S C$
We use the notation $D \in \mathcal{C}_\sim[A']$ as in Def. 4 with $\sim$ as symbol $\mathcal{R}$ there. Since $A \sim B$ we have $\mathbb{O} \neq \widehat{\sum}_{D\in\mathcal{C}_\sim[A']}T(A,a,D) = \widehat{\sum}_{D\in\mathcal{C}_\sim[A']}T(B,a,D)$ so we have $D \in \mathcal{C}_\sim[A']$ where $B\|_S C \xrightarrow{a,l} D\|_S C$ and $(A'\|_S C, D\|_S C) \in \mathcal{R}$.
<u>Case 2:</u> $C \xrightarrow{a,k} C'$, $C \not\equiv C'$ and $E \equiv A\|_S C'$
Then also $B\|_S C \xrightarrow{a,k} B\|_S C'$ and $(A\|_S C', B\|_S C') \in \mathcal{R}$.
<u>Case 3:</u> $A \xrightarrow{a,l} A$ or $C \xrightarrow{a,m} C$ and $E \equiv A\|_S C$, $k = l \mathbin{\widehat{+}} m$.
If only one of the two conditions holds, then we have the same argumentation as for case 1 or 2, respectively. If both agents contain a self loop with label a, then from $A \sim B$ we have $\mathbb{O} \neq \widehat{\sum}_{D\in\mathcal{C}_\sim[A]}T(A,a,D) = \widehat{\sum}_{D\in\mathcal{C}_\sim[A]}T(B,a,D)$. So for $D \in \mathcal{C}_\sim[A]$ we have $B \xrightarrow{a,n} D$, but since $A \in \mathcal{C}_\sim[A]$ and $A \sim B$, also $B \in \mathcal{C}_\sim[A]$, such that $(A\|_S C, B\|_S C) \in \mathcal{R}$ and

$$\widehat{\sum}_{D\in\mathcal{C}_\sim[A]}T((A\|_S C),a,(D\|_S C)) \mathbin{\widehat{+}} \widehat{\sum}_{F\in\mathcal{C}_\sim[C]}T((A\|_S C),a,(A\|_S F)) =$$

$$\widehat{\sum}_{D\in\mathcal{C}_\sim[B]}T((B\|_S C),a,(D\|_S C)) \mathbin{\widehat{+}} \widehat{\sum}_{F\in\mathcal{C}_\sim[C]}T((B\|_S C),a,(B\|_S F))$$

due to the associative and commutative law for $\widehat{+}$ in the semiring.
<u>Case 4:</u> $a \in S$, $A \xrightarrow{a,l} A'$, $C \xrightarrow{a,m} C'$ and $E \equiv A'\|_S C'$, $k = l \mathbin{\widehat{\cdot}} m$.
By $A \sim B$ we have $B \xrightarrow{a,n} B'$ with $B' \in \mathcal{C}_\sim[A']$ so clearly $B\|_S C \xrightarrow{a,o} B'\|_S C'$ and $(A'\|_S C', B'\|_S C') \in \mathcal{R}$. It remains to show equality of transition costs in the semiring. For each $\mathcal{C}_\mathcal{R} \in \mathcal{CC}_\mathcal{R}$ we have:

$$\widehat{\sum}_{(D\|_S F)\in\mathcal{C}_\mathcal{R}}T((A\|_S C),a,(D\|_S F)) = T(C,a,F) \mathbin{\widehat{\cdot}} \sum_{\mathcal{C}_\sim\in\mathcal{CC}_\sim}\widehat{\sum}_{D\in\mathcal{C}_\sim}T(A,a,D)$$
$$= T(C,a,F) \mathbin{\widehat{\cdot}} \widehat{\sum}_{\mathcal{C}_\sim\in\mathcal{CC}_\sim}\widehat{\sum}_{D\in\mathcal{C}_\sim}T(B,a,D) = \widetilde{\sum}_{(D\|_S F)\in\mathcal{C}_\mathcal{R}}T((B\|_S C),a,(D\|_S F))$$

due to distributive laws for $\widehat{+}$ and $\widehat{\cdot}$ in the semiring. This shows that the identity of transition costs holds. By a symmetric argument we complete the proof. □

It remains to show that bisimulation is a congruence for recursive behaviors. We first define under which conditions agents including variables are bisimilar.

Definition 6. *Let $A, B \in \mathcal{L}$ and let A, B include variables X_i ($i \in I$ for index set I) at most. $A \sim B$ if and only if for all agents C_i ($i \in I$)*

$$A\{C_i/X_i\} \sim B\{C_i/X_i\} \ .$$

Theorem 4. *Let $C, D \in \mathcal{L}$, $C \sim D$ and let C and D both contain variable X. Let $A = C\{A/X\}$ and $B = \{B/X\}$, then $A \sim B$.*

Proof. Since recursion by defining equations does not involve elements of the semiring, the proof follows from the proof for the Boolean model, e.g. as given in [23]. □

By means of the above theorems equivalence transformations can be defined at the syntactical level similar to the rules in process algebras like CCS [23].

5 Examples and Realizations

It is straightforward to define different process algebras using the general concept which has been presented in the previous sections. We give three application examples in the following subsections.

5.1 A Probabilistic Process Algebra

Different realizations of probabilistic processes exist in the literature [25,22]. On particular problem with probabilistic processes occurs when composition is introduced because after composition the sums of probabilities of outgoing transitions for an agent might be smaller or larger than 1. Some calculi introduce rescaling of probabilities in such a case which, however, may yield some problems with an appropriate definition of bisimulation as shown in [12]. We avoid this problem by imposing sufficient restrictions on our definition of a probabilistic process algebra for the semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$. For each agent A and each $a \in Act$ the following condition holds.

$$\sum_{B \in \mathcal{D}_a[A]} T(A, a, B) \in \{0, 1\} \tag{2}$$

Thus an agent can either perform action a or it cannot perform this action. If an action is possible, then the successor agent might be chosen probabilistically. In the notation of [25] this describes a reactive system. We assume in the sequel that *Act* does not contain label τ and all labels can be used for synchronization. Note that hiding makes it difficult to retain property (2).

Corollary 1. *Let A and B be two agents observing (2) and let Act_A and Act_B not contain label τ, then the following agents also observe (2):*

- $(a,1).A + (b,1).B$ *with* $a \neq b$,
- $(a,p).A + (a,1-p).B$ *for* $0 < p < 1$,
- $A + B$ *if* $Act_A \cap Act_B = \emptyset$,
- $A \|_S B$ *if* $(Act_A \cap Act_B) \setminus S = \emptyset$

and any combination of the above agents.

Reactive systems are usually driven by some scheduler a component which resolves the nondeterminism by choosing actions to perform. An agent AS is a scheduler if for all $A \in \mathcal{D}^*[AS]$ the following condition holds.

$$\sum_{a \in Act \setminus \{\tau\}} \sum_{B \in \mathcal{D}[A]} T(A,a,B) \in \{0,1\} \tag{3}$$

An agent $A \in \mathcal{D}^*[AS]$ is terminating if $\mathcal{D}[A] = \emptyset$. Each composition $AS\|_{Act}B$ where AS observes (3) and B observes (2) is stochastic process where the sum of outgoing probabilities is 1.0 or 0.0 for each agent. The resulting process reaches a terminating state if the scheduler reaches a terminating state and it reaches a deadlock state if it reaches a state without successors but the scheduler is not in a terminating state. Alternatively, one might be interested in an infinite behavior which means that the system never reaches a state without successors.

As a simple example we consider the well known dining philosophers problem where philosophers pick up one fork after the other, but a philosopher may first pick up the left or the right fork depending on their availability. We consider a system with N philosophers and forks where philosopher n and fork n are described by the following terms in GPA.

$$PH_n^1 = (g_n^l,1).(g_n^r,1).PH_n^2 + (g_n^r,1).(g_n^l,1).PH_n^2,$$

$$PH_n^2 = (p_n^l,1).(p_n^r,1).PH_n^1 + (p_n^r,1).(p_n^l,1).PH_n^1,$$

$$F_n = (g_n^r,1).(p_n^r,1).F_n + (g_{n+1}^l,1).(p_{n+1}^l,1).F_n$$

where $n+1$ in g_{n+1} and p_{n+1} is performed modulo N to have a cyclic system. The overall system results from composition of philosophers PH_n^1, forks F_n and a scheduler AC with $S = \cup_{i=1}^n \{g_i^l, g_i^r, p_i^l, p_i^r\}$. Two possible schedulers which allow an infinite behavior of the system are the following two.

$$\begin{aligned} AC_1 = {} & (g_1^r,1/N).(g_1^l,1).(p_1^l,1).(p_1^r,1).AC_1 + \ldots \\ & +(g_N^r,1/N).(g_N^l,1).(p_N^l,1).(p_N^r,1).AC_1 \end{aligned}$$

$$AC_2 = (g_1^l,1).(g_1^r,1).(p_1^l,1).(p_1^r,1). \;\ldots\; .(p_N^l,1).(p_N^r,1).AC_2$$

Of course, other schedulers allowing more parallelism can be defined as well.

5.2 Max/Plus Process Algebra

As a second example we consider GPA over the semiring $(\mathbb{R} \cup \{-\infty\},\ \max,\ +,\ -\infty, 0)$. To the best of our knowledge no process algebra has been proposed for this semiring yet. However, this model is well suited for the formulation of deterministic scheduling problems or several other optimization problems.

Deterministic scheduling problems are often formulated using marked graphs, a specific class of Petri nets without choices. As an alternative formalism one may as well describe these systems using GPA with the advantage of compositionality and the availability of a well defined equivalence. For the application of max/+ algebra to the analysis of timed marked graphs we refer to [3].

Max/+ algebra can only be applied for the analysis of Petri nets without choices. This restriction is, of course, taken over to GPA. Thus, an agent for the analysis of a scheduling problem must not include choices which means that the + operation cannot be used for the specification of agents.

Since the formulation of models in GPA over the max/+ semiring is straightforward, we consider a simple example. The example consists of two sources which generate parts of raw material to be assembled by a machine. The two sources are described by agent A and B and generate their material after constant times t_A and t_B.

$$A = (\tau, t_A).A_1 \quad A_1 = (a, t_p).A \quad B = (\tau, t_B).B_1 \; B_1 = (a, t_p).B$$

The parts are assembled by a machine described by component C. The machine picks up the parts if both are available by a transition with label a where picking takes time t_p for each part. It subsequently assembles both parts and offers the assembled part to some environment via a transition with label b. We assume that the machine has no intermediate buffer such that the assembled part has to be first delivered before processing of a new part can start.

$$C = (a, 0).C_1 \quad C_1 = (\tau, t_C).C_2 \quad C_2 = (b, 0).C$$

The whole system is composed of the three components. Label a is hidden because the environment interacts with the system only via b.

$$Sys = ((A \|_a B) \|_a C) \setminus \{a\}$$

The set $\mathcal{D}^*[Sys]$ contains the following 12 states which are denoted by the state of the components.

1) (A, B, C)	2) (A, B, C_1)	3) (A, B, C_2)	4) (A, B_1, C)
5) (A, B_1, C_1)	6) (A, B_1, C_2)	7) (A_1, B, C)	8) (A_1, B, C_1)
9) (A_1, B, C_2)	10) (A_1, B_1, C)	11) (A_1, B_1, C_1)	12) (A_1, B_1, C_2)

The MLTS of the system is shown in Fig. 1. In the picture, transition labels are not shown instead τ-labeled transitions are denoted by solid arcs, whereas b-labeled transitions are described by dashed arcs. Transition costs are written near the arcs. Observe that $\mathbb{O} = -\infty$ in this semiring such that all arcs which are not shown have weight $-\infty$. If $t_A = t_B$, then the relation $\mathcal{R} = \{(1), (2), (3), (4, 7), (5, 8), (6, 9), (10), (11), (12)\}$ is a bisimulation for the system and a smaller equivalent system can be generated which contains 9 instead of 12 states. Equivalent agents are surrounded by dotted boxes in the figure. The system can be analyzed according to the completion time of parts by computing vector matrix products in the general form described above.

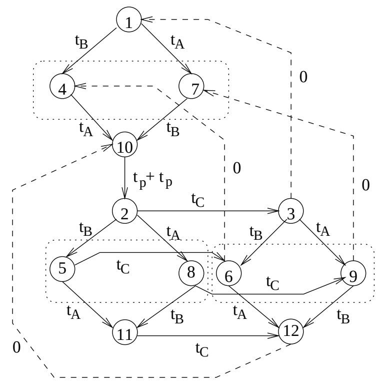

Fig. 1. MLTS of the scheduling example.

5.3 Min/Plus Process Algebra

Max/+ algebra is often used for systems with synchronization, where weights describe delays such that at a synchronization point one agent has to wait for another to do the synchronization. In min/+ algebra, $\widehat{+}$ is becomes minimum. This is useful to formulate optimization problems like shortest path problems or Markov decision problems. Of course, Min/+ and Max/+ are very similar since one can substitute the other by simply using negative values.

As an example for min/+ systems we consider a model for the computation of minimum traveling costs. The scenario is simple: A traveler wants to go from a source place to a destination place. Between source and destination several cities are located and the traveler can choose different ways and different means of transport. We use label b for buses, t for trains and c for a cab and consider as an example a route with 6 cities. City 1 is the source, city 7 is the destination which is denoted as agent 0 in our specification. The following agent defines the connections between cities.

$$\begin{array}{ll}
C_0 = (b,2).C_2 + (b,2).C_4 + (c,6).C_5 & C_1 = (b,1).C_0 + (t,2).C_5 \\
C_2 = (t,2).C_1 + (c,3).C_3 & C_3 = (c,5).0 + (t,1).C_4 + (c,4).C_5 \\
C_4 = (t,4).0 + (c,3).C_5 & C_5 = (b,2).0 + (b,2).C_3 + (t,1).C_4
\end{array}$$

A traveler can now be defined to synchronize with agent C_0. Consider first a traveler who can use arbitrary means of transport.

$$T_1 = (b,0).T_1 + (c,0).T_1 + (t,0).T_1$$

The agent $Sys_1 = (C_0 \|_{b,c,t} T_1) \setminus \{b,c,t\}$ can be used to determine the minimum traveling costs. The MLTS for this system contains 7 states. Initially $C_0 \|_{b,c,t} T$ receives cost $\mathbb{1} = 0$ and all remaining states receive costs $\mathbb{O} = \infty$. Costs can be computed by computing vectors $\mathbf{p}^k$; if $\mathbf{p}^k = \mathbf{p}^{k+1}$, then $\mathbf{p}^k(A)$ contains the minimal costs of reaching agent $A \in \mathcal{D}^*[Sys_1]$ from the initial state. The used algorithm is, of course, the well known Bellman Ford algorithm for shortest path problems [6]. Since we are interested in the costs of reaching the destination, $\mathbf{p}^k(0)$ is of interest. In this example we obtain minimal costs of 6.

We may as well define other travelers. A traveler who is not allowed to take a taxi is defined as T_2 and a traveler who wants to move at most once by train or bus is defined as T_3.

$$T_2 = (b,0).T_1 + (t,0).T_2 \qquad T_3 = (c,0).T_3 + (b,0).T_{31} + (t,0).T_{31} \quad T_{31} = (c,0).T_{31}$$

Both agents, T_2 and T_3 can be composed with C_0 like T_1. For T_2 we obtain the same minimum as for T_1, wheras T_3 has costs of 7 which is the additional price to use a cab.

6 Conclusions

In this paper we proposed a new and generic approach to define process algebras with quantified transitions. It has been shown that using an arbitrary semiring structure it is possible to define a process algebra with transition costs described by the elements of the semiring. Different behaviors of an agent are expressed by the operations of the semiring. The general process algebra GPA has been introduced based on the presented concepts. A very interesting aspect of this general approach is that a bisimulation equivalence can be defined based only on the requirements of the semiring and that the equivalence is a congruence according to the operations of the algebra.

It has been shown that by choosing concrete semirings we obtain process algebras very similar to existing process algebras. In this way it is easy to define untimed, probabilistic or stochastic algebras as it has already been done. We do not elaborate on the derivation of a stochastic algebra in our framework. Stochastic algebras mainly differ in the selection of rates in case of synchronisation [18]. Since composition is based on $\hat{\cdot}$ in GPA, selection of semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$ yields a stochastic algebra in the flavor of MTIPP [16]. Rather than deriving existing approaches, we use other semirings to achieve new process algebras including an appropriate notation of bisimulation. In this paper we present a max/+ and a min/+ process algebra which are useful for the formulation of various optimization problems. The semirings max/+ and min/+ have been considered here only for constant values. The resulting model is commonly used for the analysis of deterministic problems [3,10]. Additionally, max/+ algebra is applied for the analysis of stochastic systems [2,10] by considering non-decreasing functions instead of constants as elements of the semiring. It is obvious that most of the steps proposed in this paper can be taken over to this more general model. However, some extensions have to be introduced and are considered in future research.

References

1. R. Agrawal, F. Baccelli, and R. Rajan. An algebra for queueing networks with time varying service. Research Report 3435, INRIA, 1998.

2. F. Baccelli, G. Cohen, G. Olsder, and J. Quadrat. *Synchronization and Linearity.* John Wiley and Sons, 1992.
3. F. Baccelli, B. Gaujal, and D. Simon. Analysis of preemptive periodic real time systems using the (max,plus) algebra. Research Report 3778, INRIA, 1999.
4. F. Baccelli and D. Hong. TCP is (max/+) linear. In *Proc. SIGCOM 2000.* ACM, 2000.
5. M. Bernado and R. Gorrieri. A tuturial of EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.
6. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation.* Prentice Hall, 1989.
7. P. Buchholz. *Die strukturierte Analyse Markovscher Modelle (in German).* IFB 282. Springer, 1991.
8. P. Buchholz. Markovian process algebra: composition and equivalence. In [17].
9. P. Buchholz. Bisimulation for automata with transition costs. *submitted*, 2000.
10. C. S. Chang. *Performance guarantess in communication networks.* Springer, 1999.
11. R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: a semantics based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, 1993.
12. P. R. D'Argenio, H. Hermanns, and J. P. Katoen. On generative parallel composition. In *Proc. ProbMiv 98, Electronic Notes on Theor. Computer Sc.*, 21, 1999.
13. D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):926–944, 1984.
14. H. Hansson. Time and probability for the formal design of distributed systems. Phd thesis, University of Uppsala, 1991.
15. H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic process algebras – between LOTOS and Markov chains. *Computer Networks and ISDN Systems*, 30(9/10):901–924, 1998.
16. H. Hermanns and M. Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In [17]
17. U. Herzog and M. Rettelbach, eds. *Proc. 2nd Work. Process Algebras and Performance Modelling* Arbeitsberichte IMMD, Univ. Erlangen, Germany, no. 27, 1994.
18. J. Hillston. The nature of synchronisation. In [17].
19. J. Hillston. A compositional approach for performance modelling. Phd thesis, University of Edinburgh, Dep. of Comp. Sc., 1994.
20. J. Hillston. Compositional Markovian modelling using a process algebra. In W. J. Stewart, editor, *Computations with Markov Chains*, pages 177–196. Kluwer, 1995.
21. C. Hoare. *Communicating sequential processes.* Prentice Hall, 1985.
22. K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.
23. R. Milner. *Communication and concurrency.* Prentice Hall, 1989.
24. D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conference on Theoretical Computer Science*, pages 167–183. Springer LNCS 104, 1981.
25. R. van Glabbek, S. Smolka, B. Steffen, and C. Tofts. Reactive, generative and stratified models for probabilistic processes. In *Proc. LICS'90*, 1990.