# Priority as Extremal Probability*

Scott A. Smolka†
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400
USA
*sas@cs.sunysb.edu*

Bernhard Steffen
Lehrstuhl für Informatik II
RWTH Aachen
5100 Aachen
Germany
*bus@zeus.informatik.rwth-aachen.de*

Aachener Informatikberichte
Nr. 92 – 4

**Abstract**

We extend the stratified model of probabilistic processes presented in [vGSST90] to obtain a very general notion of *process priority*. The main idea is to allow probability guards of value 0 to be associated with alternatives of a probabilistic summation expression. Such alternatives can be chosen only if the non-zero alternatives are precluded by contextual constraints. We refer to this model as one of "extremal probability" and to its signature as $PCCS_\zeta$, where the $\zeta$ signifies the possibility of *zero*-probability alternatives. We provide $PCCS_\zeta$ with a structured operational semantics and a notion of *probabilistic bisimulation*, which is shown to be a congruence.

Of particular interest is the abstraction $PCCS_\pi$ of $PCCS_\zeta$ in which all non-zero probability guards are identified. Both the operational and bisimulation semantics of $PCCS_\zeta$ easily adapt to this abstraction. $PCCS_\pi$ represents a customized framework for reasoning about priority, and covers all features of process algebras proposed for reasoning about priority that we know of. This is illustrated by specifying a controller with a dynamic priority structure for the readers-writers problem.

*A preliminary version of this paper appeared in *Proceedings of CONCUR '90 — First International Conference on Concurrency Theory*, Vol. 458 of the Springer-Verlag series *Lecture Notes in Computer Science*, pp. 456-466, Aug. 1990.

†Research supported by NSF Grant CCR-8704309.

# 1 Introduction

In [vGSST90], three models of probabilistic processes are presented. The *reactive model*, due to Larsen and Skou [LS89], is an adaptation of the reactive viewpoint embodied in classical process algebra [Mil80,Pnu86]. A process is viewed as a black box upon which certain "button-pushing" experiments can be performed. An *$\alpha$-experiment* succeeds if the button labeled $\alpha$ goes down when pressed, in which case the process makes an $\alpha$-transition to a (possibly) new state. The experiment fails if the button will not go down, indicating that the process is in a state where it is unable to make an $\alpha$-transition. In probabilistic terms, a reactive process either possesses no $\alpha$-transitions, or else the sum of the probabilities of all $\alpha$-transitions is 1, for each action symbol $\alpha$. Restriction in the reactive model is exactly analogous to the nonprobabilistic case: restricting an action $\alpha$ will cause all subsequent $\alpha$-experiments to fail.

The *generative model* is the extension of the reactive model in which the sum of the probabilities of a state's outgoing transitions is (at most) 1. Besides encoding the relative frequency of transitions with the same label, a generative process also encodes the relative frequency of transitions with different labels. Restricting an action $\alpha$ in a generative process will cause its probability of occurrence to be distributed proportionally over the remaining transitions.

The *stratified model* is an extension of the generative model that captures the branching structure of the *purely probabilistic choices* made by a process. In contrast to the generative model where relative frequency is preserved in the context of restriction, in the stratified model restriction may alter relative frequency, albeit in a structured, level-wise way. For example, consider the process $P$ defined by the following recursive expression:

$$P \stackrel{def}{=} fix_X(\tfrac{1}{3}a\,.\,X + \tfrac{2}{3}(\tfrac{1}{2}b\,.\,X + \tfrac{1}{2}c\,.\,X))$$

$P$ may be interpreted as a scheduler that services task $a$ exactly one-third of the time, and equally services the remaining tasks ($a$ and $b$) the other two-thirds of the time. Placing $P$ in a restriction context that precludes the execution of $c$ yields process $fix_X(\frac{1}{3}a\,.\,X + \frac{2}{3}b\,.\,X)$ in the stratified model, and process $fix_X(\frac{1}{2}a\,.\,X + \frac{1}{2}b\,.\,X)$ in the generative model. Thus the above interpretation of the scheduler is retained only in the stratified model. In practice, task $a$ might correspond to a special system process, such as a garbage collector, while $b$ and $c$ would correspond to the user processes.

In this paper, we extend the stratified model to obtain a very general notion of *process priority*. This can be done simply by additionally considering the extremal case, where a probability guard is 0. For example, the expression

$$1P + 0(1Q + 0R)$$

gives priority to process $P$ over $Q$ and $R$, and priority to $Q$ over $R$. Thus process $R$ can only be executed in a restriction context that excludes $P$ and $Q$.

Our extension to the stratified model results in what we term the *model of extremal probability*. This model retains all the properties of the stratified model, and it covers all the essential features of the process algebra models for priority that we know of. Thus it provides a uniform framework in which to reason about probabilities and priorities. Also, as we shall show, a simple abstraction of the extremal probability model yields a customized framework for reasoning about priority.

Three features seem important when modeling priority. They concern the priority relation that may be defined between different actions:

1. The relation should be *globally dynamic*, i.e. it should allow an action $a$ to have priority over an action $b$ in one state of the system, and the converse in some other state.

2. It should be possible to define an arbitrary *partial order* at each system state. In particular, the relation should not be constrained to a "leveled" priority structure in which two incomparable actions are required to have the same priority relation with respect to all other actions.

3. The relation should be modeled *transparently* and *intuitively*, i.e. without using any complicated encodings.

Whereas the importance of the third feature is evident, the first two features can be illustrated and motivated by means of a "real life" example. Recall the Roman empire and their system of distributing governing power. Mainly, there were two kinds of authorities, the senate and the consuls. In times of peace, the power of the consuls was minimal. Every major decision was taken by the senate. In our interpretation this would mean that the activities of the senate are of higher priority than those of the consuls. However, in times of crisis this order was reversed: all power was transferred to the consuls because during a crisis decisions had to be made quickly, as opposed to democratically. This illustrates the necessity of having a globally dynamic priority structure—in a more technical setting, the handling of exceptions may cause a similar dynamic change in the priority structure, in order to handle run-time errors.

To illustrate the second feature, consider two Roman senators. They are independent in that neither one has priority over the other. However, if we assume that one of them is corrupted by a third party, then one of these senators is dependent on a person of which the other senator is independent. This results in a nonlinear, or non-leveled, priority structure—in a more technical setting, the handling of an exception and an interrupt may be mutually independent actions, however they serve as control entities for different types of external events.

Three central approaches to priority in process algebra are the ones by Baeten, Bergstra and Klop [BBK86], Cleaveland and Hennessy [CH90], and Camilleri and Winskel [Cam91,CW91]. Whereas the approach in [BBK86] is not globally dynamic, Cleaveland and Hennessy do not address the issue of a general partial order structure for priorities. Camilleri and Winskel satisfies both of these features. However transitions in their model must be augmented with information about the actions the environment is willing to perform in order to encode the priority structure. Thus it can be argued that their approach lacks the third feature.

Our approach satisfies all three features. It is globally dynamic, it allows for an arbitrary partial order of priorities, and it transparently models priority. The third property is achieved by explicitly representing the priority structure in terms of probability transitions. These are transitions labeled with real numbers from the interval $[0,1]$, indicating the relative frequency of choice at a certain state. The principle is the same as presented in [vGSST90]; only the handling of 0-probabilities is new. In fact, it is just this addition that provides our framework with the power to express priority.

## Summary of Technical Results

We will be working within the framework of *PCCS*, a specification language for probabilistic processes introduced in [GJS90]. *PCCS* is derived ¿from Milner's SCCS [Mil83] by replacing the operator of nondeterministic process summation with a probabilistic counterpart. Several *PCCS* expressions have appeared above, which should give the flavor of the language.

Three semantic models for *PCCS* are presented in [vGSST90]. This paper extends the stratified model $PCCS_S$ to deal with probability branches whose "a priori" probability is 0. Such a branch can be taken only if all other alternatives are precluded by context constraints. We will refer to this extension as $PCCS_\zeta$, where the $\zeta$ signifies the possibility of *zero*-probability transitions. We present the following results for $PCCS_\zeta$:

- A *structural operational semantics* is given as a set of inference rules in the style of Plotkin [Plo81] and Milner [Mil89]. These inference rules constitute a *semantic functional* from the set of $PCCS_\zeta$ expressions, $Pr_\zeta$, to a domain of probabilistic labeled transition systems.

- A notion of *bisimulation equivalence* is developed along the lines of [LS89]. There, Larsen and Skou introduced *probabilistic bisimulation*, a natural and elegant extension of strong bisimulation [Par81,Mil83]. We prove that our version of bisimulation equivalence is a congruence with respect to all $PCCS_\zeta$ operators. Thus it provides a compositional notion of semantics for $PCCS_\zeta$ that is consistent with its operational semantics.

- The effect of extending the stratified model of [vGSST90] to deal with the extremal case of zero probability guards is demonstrated by the abstraction $PCCS_\pi$ of $PCCS_\zeta$ in which all non-zero probability guards are identified. $PCCS_\pi$ represents a customized framework for reasoning about priority, and covers all essential features of process algebra models for priority that we know of.

  We present an adaptation of the operational and bisimulation semantics of $PCCS_\zeta$ to $PCCS_\pi$, and illustrate the language's utility by specifying a controller with a dynamic priority structure for the readers-writers problem.

The structure of the rest of the paper is as follows. The syntax of $PCCS_\zeta$ is presented in Section 2 and its operational and bisimulation semantics in Section 3. The language $PCCS_\pi$ is the subject of Section 4, while related work is considered in Section 5. In particular, we discuss the relationship of $PCCS_\pi$ to the approaches of Baeten, Bergstra and Klop [BBK86], Cleaveland and Hennessy [CH90], and Camilleri and Winskel [Cam91,CW91]. Conclusions and plans for future work are given in Section 6.

## 2 Syntax of PCCS$_\zeta$

As in SCCS, the *atomic actions* of $PCCS_\zeta$ constitute an Abelian monoid $(Act, \cdot, 1)$. Actions of the form $\alpha \cdot \beta$ represent the simultaneous, atomic execution by a process of the actions $\alpha$ and $\beta$. We will often use juxtaposition to denote products of actions, e.g., $\alpha\beta$. The action 1 is called the "idle action" and represents one unit of delay.

It is convenient to assume that $Act$ is generated freely from the union of the sets $\Lambda$ of *names* (e.g., $a, b, joe$) and $\overline{\Lambda}$ of *co-names* (e.g., $\overline{a}, \overline{b}, \overline{sue}$) such that:

$$\forall \alpha \in \Lambda,\ \exists\, \overline{\alpha} \in \overline{\Lambda} : \alpha \cdot \overline{\alpha} = \overline{\alpha} \cdot \alpha = 1, \text{ and vice versa.}$$

We then have that $(Act, \cdot, 1, ^{-})$ is an Abelian group.

Names and co-names play a dual role in the calculus as they denote both the *particulate actions* that processes can perform, and the *ports* where these actions can occur. The product action $\alpha \cdot \overline{\alpha} = 1$ represents a *synchronized communication* between two processes at complementary ports.

Let $X$ be a variable, $\alpha \in Act$, $A$ a subset of $Act$, and $f : Act \to Act$. Then the syntax of $PCCS_\zeta$ is given by:

$$\begin{aligned} E ::= \mathbf{0} \;\mid\; X \;\mid\; \alpha\,.\,E \;\mid\; \textstyle\sum\, [p_i]\, E_i, \;\; \text{where } p_i \in [0,1],\ \textstyle\sum p_i = 1 \\ \mid\; E \times F \;\mid\; E \upharpoonright A \;\mid\; E[f] \;\mid\; fix_X E \end{aligned}$$

An expression having no free variables is called a *process*, and $Pr_\zeta$ is the set of all $PCCS_\zeta$ processes. We let $P, Q, R$ (and occasionally $U, V, W$) range over $Pr_\zeta$. Intuitively, $\mathbf{0}$ is the *zero*

$$
\begin{array}{lcrcl}
 & & \alpha\,.\,E & \xrightarrow{\alpha} & E \\
E \xrightarrow{\alpha} E',\ F \xrightarrow{\beta} F' & \Longrightarrow & E \times F & \xrightarrow{\alpha\cdot\beta} & E' \times F' \\
E \xrightarrow{\alpha} E',\ \alpha \in A & \Longrightarrow & E \lceil A & \xrightarrow{\alpha} & E' \lceil A \\
E \xrightarrow{\alpha} E' & \Longrightarrow & E\,[f] & \xrightarrow{f(\alpha)} & E'\,[f] \\
E\{fix_X E/X\} \xrightarrow{\alpha} E' & \Longrightarrow & fix_X E & \xrightarrow{\alpha} & E'
\end{array}
$$

Figure 1: Inference Rules for Action Transitions of $PCCS_\zeta$

*process* having no transitions, while $\alpha\,.\,E$ performs action $\alpha$ with probability 1 and then behaves like $E$. The expression $\sum[p_i]\,E_i$ offers a probabilistic choice among its constituent behaviors $E_i$, while $E \times F$ represents synchronized product. The restricted expression $E\lceil A$ can only perform actions from the set $A$, and $E[f]$ specifies a relabeling of actions. Finally, $fix_X E$ defines a recursive process.

In this paper, we require summation expressions to be finite, assume only guarded recursion, write the binary version of summation as $[p]\,E + [1-p]\,F$, and often omit the square brackets around the probabilities.

The only difference between the syntax of $PCCS_\zeta$ and the syntax of $PCCS$ presented in [vGSST90] is the consideration of 0-probability guards in summation expressions. The effect of restriction is a normalization that proportionally distributes the probabilities of the restricted transitions over the remaining transitions. This requires a somewhat arbitrary decision when normalizing a process like $0\,P + 0\,Q$, which we resolve by distributing the probabilities evenly.

# 3 The Extremal Probability Model

In this section, we first lift the operational semantics of the stratified model of probabilistic processes to obtain the *extremal probability model*. This is accomplished by augmenting the range of normalized probabilities with the bottom element $\perp$. The presence of $\perp$ permits the distinction between a restricted process that has no permitted transitions, such as $(\frac{1}{3}\,a.\mathbf{0} + \frac{2}{3}\,b.\mathbf{0})\lceil\{c\}$; versus a restricted process that does have permitted action transitions but with accompanying probability transitions of 0, e.g. $(1\,a.\mathbf{0} + 0\,b.\mathbf{0})\lceil\{b\}$. In our semantics, the former process will be equivalent to $\mathbf{0}$ and the latter process will be equivalent to $1\,b.\mathbf{0}$. We then equip the extremal probability model with a notion of probabilistic bisimulation, and show that the resulting equivalence relation is a congruence with respect to $PCCS_\zeta$.

## 3.1 Operational Semantics of PCCS$_\zeta$

As for the stratified model, the structural operational semantics of $PCCS_\zeta$ in the extremal probability model is given in terms of inference rules that define two types of transition relations: *action transitions*, as in SCCS, and *probability transitions*. Action transitions are of the form $P \xrightarrow{\alpha} Q$. Probability transitions are of the form $P \overset{p}{\longmapsto} Q$, meaning that $P$, with probability $p$, can behave as process $Q$. This separation of action and probability permits the branching structure of probabilistic choices to be captured explicitly. The rules for action transitions are given in Figure 1. They are the same as in SCCS [Mil83] with the exception that there is no rule for process summation—in the stratified and extremal probability models, the only choice

$$
\begin{array}{llll}
 & & \sum_i [p_i]E_i \;\stackrel{p_i}{\longmapsto}_i & E_i \\
E \stackrel{p}{\longmapsto}_i E',\ F \stackrel{q}{\longmapsto}_j F' & \Longrightarrow & E \times F \;\stackrel{p\cdot q}{\longmapsto}_{(i,j)} & E' \times F' \\
E \stackrel{p}{\longmapsto}_i E',\ F \stackrel{\alpha}{\longrightarrow} F' & \Longrightarrow & E \times F \;\stackrel{p}{\longmapsto}_{(i,0)} & E' \times F \\
E \stackrel{\alpha}{\longrightarrow} E',\ F \stackrel{p}{\longmapsto}_i F' & \Longrightarrow & E \times F \;\stackrel{p}{\longmapsto}_{(0,i)} & E \times F' \\
E \stackrel{p}{\longmapsto}_i E',\ \nu_\zeta(E',A) \neq \bot & \Longrightarrow & E\lceil A \;\stackrel{\rho_\zeta(E,A,p)}{\longmapsto}_i & E'\lceil A \\
E \stackrel{p}{\longmapsto}_i E' & \Longrightarrow & E[f] \;\stackrel{p}{\longmapsto}_i & E'[f] \\
E\{fix_X E/X\} \stackrel{p}{\longmapsto}_i E' & \Longrightarrow & fix_X E \;\stackrel{p}{\longmapsto}_i & E'
\end{array}
$$

Figure 2: Inference Rules for Probability Transitions of $PCCS_\zeta$

mechanism is probabilistic. The inference rules for probability transitions appear in Figure 2.

Except for the third, fourth, and fifth rules, all of the inference rules of Figure 2 are straightforward adaptations of the standard SCCS rules. (Another difference from SCCS is that probability transitions are indexed, a point we discuss below.) The third and fourth rules are needed to avoid deadlock in a synchronous product that is caused by a difference in depth of the purely probabilistic branching structures of the argument processes. For example, we do not want $(\frac{1}{2}\, a.\mathbf{0} + \frac{1}{2}\, b.\mathbf{0}) \times c.\mathbf{0}$ to deadlock simply because there does not exist a probability transition in the right argument. This is avoided by the third rule. The fourth rule is symmetric.

The fifth rule deals with the restriction operator, and expresses the probability transitions of $E\lceil A$ in terms of conditioned probability transitions of $E$. Intuitively, $E\lceil A$ behaves like $E$, where all probability transitions to subexpressions that necessarily require the execution of a restricted action are eliminated. The probabilities associated with these transition are proportionately distributed among the remaining probability transitions at the same level of probabilistic choice. As discussed in greater detail below, the second hypothesis of the restriction rule is what realizes this pruning away of probability transitions.

The term $\rho_\zeta(E,A,p)$ gives the conditional probability of the derived transition for expression $E$ in a restriction context that permits only actions from the set $A$. It is defined by

$$
\rho_\zeta(E,A,p) = \begin{cases} \bot & \text{if } \nu_\zeta(E,A) = \bot \\ \frac{1}{n} & \text{if } \nu_\zeta(E,A) = 0 \text{ and } n = |\{E \stackrel{0}{\longmapsto}_i E_i \mid \nu_\zeta(E_i,A) \neq \bot\}| \\ p/\nu_\zeta(E,A) & \text{otherwise} \end{cases}
$$

Here $\nu_\zeta(E,A)$ is the "normalization factor" of expression $E$ in restriction context $A$, and is defined formally below. A normalization factor of $\bot$ (the first clause of $\rho_\zeta$) indicates that the restricted expression possesses no transitions, and $\rho_\zeta$ is also defined to be $\bot$ in this case. By the second hypothesis of the restriction rule, $\rho_\zeta$ is only applied in situations where $\nu_\zeta(E',A) \neq \bot$, which implies $\nu_\zeta(E,A) \neq \bot$ also. Therefore, the purpose of the first clause is merely to provide a complete and consistent definition of $\rho_\zeta$.

A normalization factor of 0 (the second clause of $\rho_\zeta$) means that only probability-0 transitions remain unrestricted and that there is at least one of them. Let there be $n \geq 1$ of these transitions. Then they are transformed into probability-$\frac{1}{n}$ transitions, thus enabling the zero-probability portion of the process for execution. The second clause therefore yields equations like

$$
(1\, a.P \,+\, 0\, b.Q \,+\, 0\, c.R)\lceil\{b,c\} \;=\; \tfrac{1}{2}\, b.Q \,+\, \tfrac{1}{2}\, c.R
$$

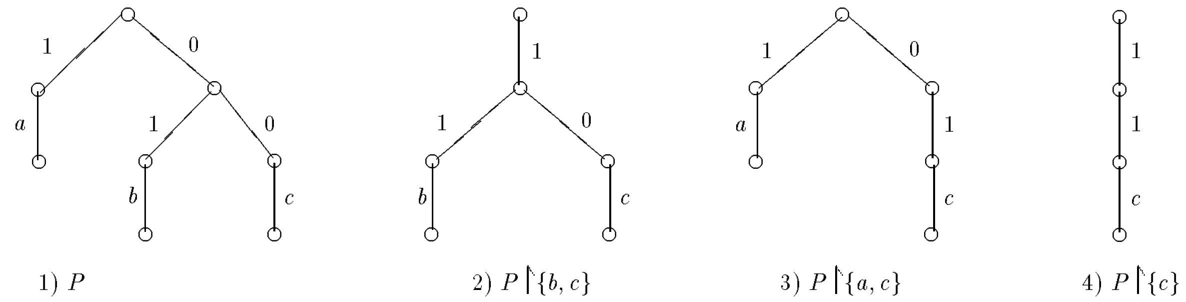

Figure 3: Extremal Probability Transition Systems

meaning that processes of lower priority may only be executed when all processes of higher priority are excluded by the context.

The final clause of $\rho_\zeta$ corresponds to a value for $\nu_\zeta(E, A)$ in the interval $(0, 1]$, which is used to compute the conditional probabilities of the transitions of $E \lceil A$ under the assumption of $A$. For example,

$$(\tfrac{1}{3}\, a.P \,+\, \tfrac{1}{3}\, b.Q \,+\, \tfrac{1}{3}\, c.R) \lceil \{a, b\} \;=\; \tfrac{1}{2}\, a.P \,+\, \tfrac{1}{2}\, b.Q$$

Using $\{\!|,|\!\}$ as multi-set brackets, and adopting the convention that the empty summation is $\perp$ (and not 0 as usual), the formal definition of $\nu_\zeta$ is given by:

$$\nu_\zeta(E, A) = \begin{cases} 1 & \text{if } E \xrightarrow{\alpha}, \alpha \in A \\ \perp & \text{if } E \xrightarrow{\beta}, \beta \notin A \\ \sum_i \{\!|\, p_i \mid E \overset{p_i}{\longmapsto}_i E_i \;,\; \nu_\zeta(E_i, A) \neq \perp |\!\}, & \text{otherwise} \end{cases}$$

The condition $\nu_\zeta(E', A) \neq \perp$ in the premise of the restriction inference rule means that derivative $E'$ of $E$ is capable of performing an action transition from the set $A$, perhaps with probability 0. Consequently, the probability transition of the premise should remain after restriction, conditioned under the assumption of $A$.

To illustrate both the inference rule for restricted expressions and the way in which priority is captured consider the process

$$P \overset{def}{=} 1\, a.\mathbf{0} \,+\, 0\,(1\, b.\mathbf{0} \,+\, 0\, c.\mathbf{0})$$

In the following, $P$ is "partially evaluated" with respect to a representative sampling of the relevant restriction contexts. The results of these evaluations are the restriction-free expressions on the right-hand side.

$$\begin{array}{lll} 1) & P \lceil \{a, b, c\} & \leadsto \; 1\, a.\mathbf{0} \,+\, 0\,(1\, b.\mathbf{0} \,+\, 0\, c.\mathbf{0}) \\ 2) & P \lceil \{b, c\} & \leadsto \; 1\,(1\, b.\mathbf{0} \,+\, 0\, c.\mathbf{0}) \\ 3) & P \lceil \{a, c\} & \leadsto \; 1\, a.\mathbf{0} \,+\, 0\, 1\, c.\mathbf{0} \\ 4) & P \lceil \{c\} & \leadsto \; 1\,(1\, c.\mathbf{0}) \\ 5) & P \lceil \{\,\} & \leadsto \; \mathbf{0} \end{array}$$

Further evaluation of these expressions would yield the stratified transition systems of Figure 3.

As mentioned above, all probability transitions are indexed. The indices are used to distinguish different *occurrences* of the *same* probability transition, and are constructed so that every probability transition of an expression has a unique index. (The indices were used above in computing the normalization factor of a restricted expression, and will be used in the next section to define *cumulative* probability distributions.) The following examples are illustrative:

$$(\tfrac{1}{2}\, a\,.\,\mathbf{0} + \tfrac{1}{2}\, a\,.\,\mathbf{0}) \overset{\frac{1}{2}}{\longmapsto}_{1} a\,.\,\mathbf{0} \qquad (\tfrac{1}{2}\, a\,.\,\mathbf{0} + \tfrac{1}{2}\, a\,.\,\mathbf{0}) \overset{\frac{1}{2}}{\longmapsto}_{2} a\,.\,\mathbf{0}$$

$$(\tfrac{2}{3}\, a\,.\,\mathbf{0} + \tfrac{1}{3}\, b\,.\,\mathbf{0}) \times c\,.\mathbf{0} \overset{\frac{1}{3}}{\longmapsto}_{(2,0)} b\,.\mathbf{0} \times c\,.\mathbf{0}$$

The inference rules of Figures 1 and 2 define the semantics of $PCCS_\zeta$ expressions in terms of transition systems ¿from the extremal probability model. Such transition systems are *stochastic* in the sense that for each state with one or more outgoing probability transitions, the sum of the probabilities of these transitions is 1.

## 3.2 Bisimulation Semantics

Bisimulation in the extremal probability model can now be defined in a fashion analogous to stratified bisimulation [vGSST90]. As prerequisites, we need to define $\mu_\zeta$, the *cumulative probability distribution function* (PDF), which computes the total probability by which a process derives a set of processes. $\mu_\zeta$ is of the form

$$\mu_\zeta : (Pr_\zeta \times (Act \ \cup \ \{\,*\,\}) \times \mathcal{P}(Pr_\zeta)) \longrightarrow [0,1] \cup \{\bot\}$$

where $\mathcal{P}(Pr_\zeta)$ denotes the power set of $Pr_\zeta$, and $*$ is a dummy symbol used to mark probability transitions. For $\alpha \in Act$, $\mu_\zeta(P, \alpha, S) \in \{\,1, \bot\,\}$ indicates whether or not $P$ has an $\alpha$–transition to some process in $S$. Otherwise, $\mu_\zeta(P, *, S) \in [0,1] \cup \{\bot\}$ indicates the total probability by which $P$ can behave as some process in $S$. The case of $\bot$ indicates that $P$ possesses no probability transition to any process in $S$. Recalling that the empty sum of probabilities is $\bot$, we have:

**Definition 1 (Cumulative PDF)** $\mu_\zeta$ *:* $(Pr_\zeta \times (Act \ \cup \ \{\,*\,\}) \times \mathcal{P}(Pr_\zeta)) \longrightarrow [0,1] \cup \{\bot\}$ *is the total function given by :* $\forall \alpha \in Act \cup \{\,*\,\}, \forall\, P \in Pr_\zeta, \ \forall\, S \subseteq Pr_\zeta$*.*

$$\mu_\zeta(P, \alpha, S) = \begin{cases} 1 & \textit{if } \ \exists Q \in S.\ P \overset{\alpha}{\longrightarrow} Q \\ \sum \{\!|\, p_i \mid Q \in S\,,\ P \overset{p_i}{\longmapsto}_i Q |\!\} & \textit{if } \ \alpha = * \\ \bot & \textit{otherwise} \end{cases}$$

Given an equivalence relation $\mathcal{R}$ on $Pr_\zeta$, we denote its corresponding set of equivalence classes by $Pr_\zeta/\mathcal{R}$ and define:

**Definition 2** *An equivalence relation* $\mathcal{R} \subseteq Pr_\zeta \times Pr_\zeta$ *is a* $\zeta$-bisimulation *if* $(P,Q) \in \mathcal{R}$ *implies* $\forall S \in Pr_\zeta/\mathcal{R}, \ \forall \alpha \in Act \cup \{\,*\,\}$*,*

$$\mu_\zeta(P, \alpha, S) = \mu_\zeta(Q, \alpha, S)$$

*Two processes* $P, Q$ *are* $\zeta$-bisimulation equivalent *(written* $P \overset{\zeta}{\sim} Q$*) if there exists a* $\zeta$*-bisimulation* $\mathcal{R}$ *such that* $(P,Q) \in \mathcal{R}$*.*

If $\mathcal{R}_1$ and $\mathcal{R}_2$ are $\zeta$-bisimulations, then the transitive closure of their union $(\mathcal{R}_1 \cup \mathcal{R}_2)^*$ is again a $\zeta$-bisimulation. Thus, as in the case of classical bisimulation, $\zeta$-bisimulation equivalence is the largest $\zeta$-bisimulation, *i.e.*

$$\overset{\zeta}{\sim} \ = \ \bigcup \{\ \mathcal{R} \mid \mathcal{R} \text{ is a } \zeta\text{-bisimulation}\ \}$$

and can be found by a straightforward adaptation of the fixed-point iteration technique of [Mil89].

Like strong bisimulation does for SCCS or CCS, $\zeta$-bisimulation equivalence provides a compositional notion of semantics for $PCCS_\zeta$ that is consistent with the operational semantics defined in the last section. Specifically:

**Theorem 1** (**Congruence**) *For* $P, Q \in Pr_\zeta$*:*

$$P \overset{\zeta}{\sim} Q \ \textit{ implies for all } PCCS_\zeta \textit{ contexts } \mathcal{C}[\,]: \quad \mathcal{C}[P] \overset{\zeta}{\sim} \mathcal{C}[Q]$$

**Proof:** The proof proceeds by induction on the structure of contexts. The case for the empty environment is trivial. Thus it only remains to investigate the cases corresponding to the top-most operator of a context. We present the cases of product and restriction here; the other cases are similar (or even easier). For an equivalence relation $\mathcal{R}$ and a process $P$, we write $[P]_\mathcal{R}$ to denote the equivalence class of which $P$ is a member.

**Product:**

We show that $P \overset{\zeta}{\sim} Q$ implies $P \times R \overset{\zeta}{\sim} Q \times R$. Assume that $(P, Q)$ belongs to the $\zeta$-bisimulation $\mathcal{R}_0$. It is sufficient to show that the relation

$$\mathcal{R} = \{\, (U \times W, V \times W) \mid (U, V) \in \mathcal{R}_0, W \in Pr_\zeta \,\} \quad \cup \quad ID_{Pr_\zeta}$$

where $ID_{Pr_\zeta}$ is the identity relation over $Pr_\zeta$, is a $\zeta$-bisimulation. First note that since $\mathcal{R}_0$ is an equivalence relation, then so is $\mathcal{R}$. Now suppose $(U \times W, V \times W) \in \mathcal{R}$, and let $r = \mu_\zeta(U \times W, \gamma, [U' \times W']_\mathcal{R})$ $(= \mu_\zeta(U \times W, \gamma, [U']_{\mathcal{R}_0} \times W')\,)$. Then there are two cases to be distinguished.

$\gamma \in Act$ :

In this case we have

$$r = \mu_\zeta(U, \alpha, [U']_{\mathcal{R}_0}) = \mu_\zeta(U, \beta, U') = 1\,, \ \text{ and } \ \gamma = \alpha \cdot \beta$$

for unique actions $\alpha$ and $\beta$. Thus, $U \overset{\zeta}{\sim} V$ yields $\mu_\zeta(V, \alpha, [U']_{\mathcal{R}_0}) = 1$, and therefore $\mu_\zeta(V \times W, \gamma, [U' \times W']_\mathcal{R}) = 1$ as desired.

$\gamma = *$ :

In this case we have

$$\mu_\zeta(U, *, [U']_{\mathcal{R}_0}) = p, \ \ \mu_\zeta(U, *, U') = q\,, \ \text{ and } \ r = pq$$

for unique $p, q \in [0, 1]$. Thus $U \overset{\zeta}{\sim} V$ yields $\mu_\zeta(V, *, [U']_{\mathcal{R}_0}) = p$, and therefore $\mu_\zeta(V \times W, *, [U' \times W']_\mathcal{R}) = r$, which completes the proof in this case.

The argument that pairs of the form $(U, U)$ belong in $\mathcal{R}$ is trivial, and thus $\mathcal{R}$ is a $\zeta$-bisimulation. The symmetric result that $P \overset{\zeta}{\sim} Q$ implies $R \times P \overset{\zeta}{\sim} R \times Q$, which can be proven exactly in the same way, completes the proof of the case of product.

**Restriction:**

Here we must show that $P \overset{\zeta}{\sim} Q$ implies $P \lceil A \overset{\zeta}{\sim} Q \lceil A$, for $A \subseteq Act$. Thus let $(P, Q)$ belong to the $\zeta$-bisimulation $\mathcal{R}_0$. It is enough to show that the relation

$$\mathcal{R} = \{ (U \lceil A, V \lceil A), \mid (U, V) \in \mathcal{R}_0 \} \quad \cup \quad ID_{Pr_\zeta}$$

is a $\zeta$-bisimulation. This follows straightforwardly from the definition of restriction and the following fact:

$$U \overset{\zeta}{\sim} V \text{ implies } \forall A \subseteq Act : \nu_\zeta(U, A) = \nu_\zeta(V, A)$$

This fact can be proved by case analysis on the definitions of $\nu_\zeta$ and $\mu_\zeta$. $\Box$

# 4 PCCS$_\pi$ – A Calculus of Priority

In this section we consider the language $PCCS_\pi$, the abstraction of $PCCS_\zeta$ where all non-zero probability guards are identified. To make this explicit in the syntax, we denote all such guards uniformly as 1. The idea is that all non-zero probability guarded alternatives have priority over zero probability guarded alternatives, but are incomparable among themselves. Combining this with the hierarchical nature of the stratified model yields a very general framework for reasoning about priority.

## 4.1 Syntax and Semantics of PCCS$_\pi$

The syntax of $PCCS_\pi$ is therefore the same as that of $PCCS_\zeta$ except that "probability guards", from now on referred to as "priority guards", are chosen from the set $\{0, 1\}$ rather than the interval $[0, 1]$. Also the side condition that the priority guards of a summation expression sum to 1 is replaced by the requirement that any $PCCS_\pi$ summation must contain at least one summand guarded by 1. Furthermore, the inference rules for action transitions and "probability transitions", the latter from now on referred to as "priority transitions", are the same as in $PCCS_\zeta$ (Figures 1 and 2, respectively). Only the manner in which the conditional probabilities of a restricted expression are computed changes. The new, simplified, definition is as follows:

$$\rho_\pi(E, A, p) = \begin{cases} \perp & \text{if } \nu_\zeta(E, A) = \perp \\ 1 & \text{if } \nu_\zeta(E, A) = 0 \\ p & \text{otherwise} \end{cases}$$

The first clause is the same as before, while the second deals with the case where only priority-0 transitions remain after restriction. Under these circumstances, the priority-0 transitions are "lifted to the surface" by transforming them into priority-1 transitions. Finally, the third clause deals with case where at least one priority-1 transition remains unrestricted. In this case the original priority is maintained.

Note that we do not have to change the definition of the normalization factor from the extremal probability model ($\nu_\zeta$) in the definition of $\rho_\pi$. This is true even though the range of $\nu_\zeta$ on $PCCS_\pi$ expressions is the natural numbers union $\{\perp\}$ (a $PCCS_\pi$ expression may have an arbitrary number of priority-1 guards and an arbitrary number of them may be summed together in the computation of $\nu_\zeta$). This has no bearing on the computation of $\rho_\pi$ as there are only three cases of interest, $\nu_\zeta = \perp$ or 0, or $\nu_\zeta \geq 0$.

To illustrate the new situation let us consider the following example

$$P \stackrel{def}{=} 1\,a.\mathbf{0} \,+\, 1\,b.\mathbf{0} \,+\, 0\,(1\,c.\mathbf{0} \,+\, 0\,d.\mathbf{0} \,+\, 0\,e.\mathbf{0})$$

in some example contexts. The resulting "partial evaluation" into restriction-free expressions should clarify the roles of multiple 1 and 0 priority alternatives in a $PCCS_\pi$ summation expression.

$$\begin{array}{lll}
1) & P\lceil\{b,c,d,e\} & \leadsto \; 1\,b.\mathbf{0} \,+\, 0\,(1\,c.\mathbf{0} \,+\, 0\,d.\mathbf{0} \,+\, 0\,e.\mathbf{0}) \\
2) & P\lceil\{c,d,e\} & \leadsto \; 1\,(1\,c.\mathbf{0} \,+\, 0\,d.\mathbf{0} \,+\, 0\,e.\mathbf{0}) \\
3) & P\lceil\{b,d,e\} & \leadsto \; 1\,b.\mathbf{0} \,+\, 0\,(1\,d.\mathbf{0} \,+\, 1\,e.\mathbf{0}) \\
4) & P\lceil\{d,e\} & \leadsto \; 1\,(1\,d.\mathbf{0} \,+\, 1\,e.\mathbf{0})
\end{array}$$

For the bisimulation semantics we need only change the second clause of the $\mu_\zeta$ (Definition 1) to obtain $\mu_\pi$, the cumulative PDF for $PCCS_\pi$. The new definition is:

$$\mu_\pi(P,\alpha,S) = \begin{cases} 1 & \text{if } \exists Q \in S.\, P \xrightarrow{\alpha} Q \\ max\,\{p_i \,|\, Q \in S\,,\; P \overset{p_i}{\longmapsto}_i Q\} & \text{if } \alpha = * \\ \perp & \text{otherwise} \end{cases}$$

Like before, we adopt the convention that $max$ of the empty set is $\perp$. Thus the range of $\mu_\pi$ is simply $\{0,1,\perp\}$. Note that the second clause of $\mu_\pi$ uses sets rather than the multisets appearing in the definition of $\mu_\zeta$. This is because we no longer need to distinguish between different occurrences of similar priority transitions (cf. Definition 1).

The definition of a $\pi$-bisimulation is obtained from Definition 2 simply by using $\mu_\pi$ instead of $\mu_\zeta$. Now $\stackrel{\pi}{\sim}$ is the largest $\pi$-bisimulation. We also have that $\pi$-bisimulation is a congruence with respect to the operators of $PCCS_\pi$. The proof is similar to that of Theorem 1.

### 4.2 Example: Readers-Writers Problem

We illustrate the utility of $PCCS_\pi$ by considering the well-known readers-writers problem, in which reader processes and writer processes are vying for access to shared data. Any number of readers may be granted simultaneous access to the data but a writer must access the data in a mutually exclusive fashion.

We describe a solution to the problem in which writers have priority over readers: intuitively, a request by a writer for access to the data must be granted as soon as possible. In more concrete terms, once a writer is waiting, no more readers will be allowed to start reading. Moreover, our solution possesses a dynamic priority structure in which the urgency of a writer's request increases with the number of readers in their reading stage.

Our solution is given in terms of a centralized controller process that handles access requests from readers and writers. Readers go through the protocol:

$$Reader = \overline{sr}\,.\,read\,.\,\overline{er}\,.\,Reader$$

where $sr$ stands for "start read" and $er$ for "end read". Writers follow the protocol:

$$Writer = \overline{req}\,.\,\overline{sw}\,.\,write\,.\,\overline{ew}\,.\,Writer$$

$$
\begin{array}{lll}
<0,F> = & 1\,req\,.<0,T> \;+\; 1\,sr\,.<1,F> & \\
<0,T> = & sw\,.ew\,.<0,F> & \\
<i,T> = & er\,.<i-1,T> & i>0 \\
<i,F> = & 1\,er\,.<i-1,F> \;+\; 1\,sr\,.<i+1,F> \;+\; 1\,req\,.<i,T> & 0<i<m \\
<i,F> = & 1\,(1\,er\,.<i-1,F> \;+\; 0\,sr\,.<i+1,F>) + & m \leq i < n \\
 & \qquad 1\,(1\,req\,.<i,T> \;+ 0\,sr\,.<i+1,F>) & \\
<i,F> = & 1\,req\,.<i,T> \;+\; 0\,(1\,er\,.<i-1,F> \;+\; 0\,sr\,.<i+1,F>) & i \geq n
\end{array}
$$

Figure 4: $PCCS_\zeta$ specification of the controller for the readers-writers problem

where *req* stands for "request to write", *sw* for "start write", and *ew* for "end write".

We use $PCCS_\pi$ to specify three levels of urgency in the controller, which in a typical state must choose between starting a reader (*sr*), receiving (and eventually granting) a request from a writer (*req*), and ending a reader (*er*). Note, however, that once the controller has received a *req* message, it thereafter will accept only *er* messages until no readers are left. At this point, the writer gains access to the data.

When $\#readers$, the number of readers in their reading stage, is such that $0 \leq \#readers < m$, then no priority relation is specified among the three actions *sr*, *req*, and *er*. When $m \leq \#readers < n$ the controller gives both *er* and *req* priority over *sr*. Finally, when $\#readers \geq n$ the controller gives *req* priority over *er*, and *er* priority over *sr*.

The $PCCS_\pi$ specification of the controller is given in Figure 4 and a graphical depiction of the specification appears in Figure 5. States of the specification are of the form $<i,T>$ or $<i,F>$, $i \geq 0$. The first type corresponds to a state in which $i$ readers are in their reading stage and the controller has received a request from a writer that it needs to act upon. The second type corresponds to a state in which $i$ readers are in their reading stage and the controller has granted all writer requests that it has received. The initial state of the controller is $<0,F>$.

# 5 Related Work

In the following, we will concentrate on $PCCS_\pi$ and its relationship to the approaches of Baeten, Bergstra and Klop [BBK86], Cleaveland and Hennessy [CH90], and Camilleri and Winskel [Cam91,CW91]. Whereas their approaches are based on asynchronous languages, our approach is based on the synchronous language SCCS. This difference does not play a prominent role in our comparison of the approaches. Instead, we will concentrate on the way priority is modeled, in particular, with respect to the three features introduced in Section 1.

### The Baeten-Bergstra-Klop Approach

The approach of Baeten, Bergstra and Klop [BBK86], which we will henceforth refer to as the BBK-approach, is based on a partial ordering $<$ on atomic processes (*i.e.* actions). This partial order globally expresses priority *potential.* That is, at every state there are two possibilities: either, the priority relation is the one expressed by the partial order on the action set, which is the case if the state occurs within the scope of the "priority evaluator" $\theta$, or no priority structure is imposed. Thus, the BBK-approach is static in that the same priority relation on actions holds at all possible states. For example, if $a < b$ at some state of the system, then $a > b$ cannot be valid at any other state.

The BBK-approach therefore satisfies the second of the three features we have proposed,

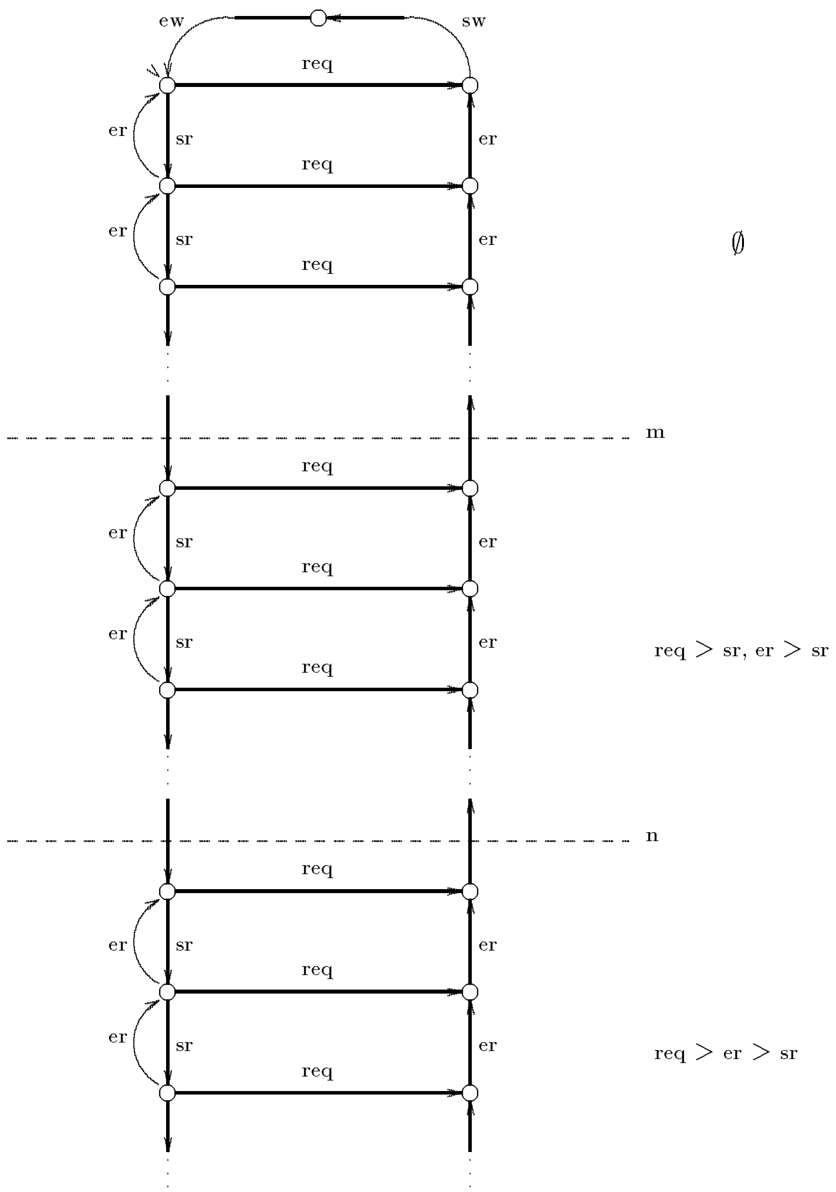


Figure 5: Structure of the controller for the readers-writers problem.

but not the first. Finally, there is a significant difference between expressing priority potential, as is done in the BBK-approach, versus explicitly specifying a process that consists of two subprocesses with one having priority over the other. Thus the third feature, which concerns the transparent modeling of priority, rather than priority potential, does not apply to the BBK-approach.

### The Cleaveland-Hennessy Approach

The approach of Cleaveland and Hennessy [CH90], which we will refer to as the CH-approach, is based on the idea of providing a prioritorized version $\underline{a}$ and a standard version $a$ for each action $a \in Act$. This allows the priority relation between actions $a, b \in Act$ to vary from one state to another by writing $\underline{a} + b$ at one state, and $a + \underline{b}$ at another. In this respect, the CH-approach is more general than the BBK-approach. However, it is less general in terms of the priority relations that can be expressed. In particular, the CH-approach distinguishes between two categories of actions, prioritorized and unprioritorized. As pointed out in [CH90], this approach can be easily extended to a multi-level priority structure. However an extension to the generality of the BBK-approach, where arbitrary partial orders are allowed, is less obvious.

An example of such a general priority relation is given by $a > b$ and $a > c$ and $d > c$. Here $a$ has priority over both $b$ and $c$, and $d$ has priority only over $c$. To express partial orders of this kind, the CH-approach would require a rather complicated structure on its action set involving a different set of indices for each potential partial order. Our approach naturally covers such general priority relations. For example the priority relation defined above could be expressed by: $1\,(1\,a + 0\,b) + 1\,(1\,(1\,a + 1\,d) + 0\,c)$. In summary, the CH-approach possesses the first and the third feature, but not the second.

### The Camilleri-Winskel Approach

In their process algebra, Camilleri and Winskel [Cam91,CW91] express priority in essentially the same way as do we; *i.e.* ignoring details, $P \nrightarrow Q$ and $1\,P + 0\,Q$ have the same meaning. Thus, the Camilleri-Winskel approach has the same degree of generality as ours: it is globally dynamic and at each state the priority structures may constitute an arbitrary partial order.

There is, however, a significant difference between Camilleri and Winskel's approach and ours at the semantic model (transition system) level. Whereas we simply maintain the priority structure in terms of probability (priority) transitions, Camilleri and Winskel indirectly express this structure essentially by recording all context-transitions pairs. More technically, transitions are augmented with *ready sets* in their model, which characterize the nature of the contexts in which a transition is possible.

### Recent Work on Priority

A brief description of several recent approaches to priority in process algebra is now given. Gerber and Lee [GL90] present a process calculus that explicitly takes into account the availability of "resources" (e.g., processors) on which processes can execute: true concurrency is possible between processes executing on distinct resources, while processes executing on the same resource are interleaved. A static priority structure is defined over resources and an accompanying notion of prioritorized strong bisimulation is given.

Another approach dealing with true concurrency is that by Best and Koutny [BK90] which investigates a new semantics for static priorities within the framework of Petri Nets.

Tofts encodes priority by means of higher ordinals: a transition has priority over another if their weights are separated at least by a factor of $\omega$. Thus all finitely separated weights yield the same priority. In comparison to a probability-based approach, this technique simplifies the definition of the operational semantics as no normalization is needed. Normalization, however, cannot be avoided entirely; it is essentially only delayed until the definition of bisimulation. We prefer to normalize at the operational level, because of the low-level (operational, if you will) nature of the normalization process, and it also results in a simpler definition of bisimulation. Moreover, we consider the probabilistic approach as more intuitive than the weight-based approach.

German [Ger91] introduces a CCS-like language in which negation and conjunction are defined over the space of atomic actions. This permits the specification of multi-process synchronization and priority within a unified framework. For example, to give priority to action $a$ over action $b$ one can write $a + (b \wedge \neg a)$ meaning that $b$ can occur only when $a$ is not possible. Using this technique a dynamic and general partial order priority structure can be expressed.

## 6 Conclusions

We have presented a language that uniformly captures both probability and priority in a very general way. This has been achieved simply by extending the syntax of the language $PCCS_S$ to allow probability guards of 0, in conjunction with a careful treatment of 0-probability transitions at the semantic level. The expressive power of the resulting language $PCCS_\zeta$ has been demonstrated: already a strong simplification of $PCCS_\zeta$ covers the essential features of the known process algebra models for priority.

One can easily obtain an asynchronous version of $PCCS_\pi$. A suitable asynchronous parallel operator would interleave priority transitions and deal with action transitions like e.g. CCS. Another choice would be to require some synchronization on priority transitions. We are currently investigating the first alternative in more detail.

**Acknowledgements:** The authors would like to thank Ivan Christoff and Chris Tofts for valuable discussions on probability and priority in process algebra.

## References

[BBK86] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae IX*, 127–168, 1986.

[BK90] E. Best and M. Koutny. *Partial Order Semantics of Priority Systems.* Technical Report 6/90, Institute of Computer Science, University of Hildesheim, Hildesheim, Germany, June 1990.

[Cam91] J. A. Camilleri. *Priority in Process Calculi.* Technical Report No. 227, Computer Laboratory, Univesity of Cambridge, Ph.D. Thesis, Cambridge, England, June 1991.

[CH90] R. J. Cleaveland and M. C. B. Hennessy. Priorities in process algebras. *Information and Computation*, 87(1/2):58–77, 1990.

[CW91] J. A. Camilleri and G. Winskel. CCS with priority choice. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, Amsterdam, July 1991.

[Ger91] S. M. German. *A Language for Specifying Synchronization.* Technical Report, GTE Laboratories, Inc., Waltham, MA, 1991.

[GJS90] A. Giacalone, C.-C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods*, IFIP TC 2, Sea of Gallilee, Israel, April 1990.

[GL90] R. Gerber and I. Lee. A calculus for communicating shared resources. In *CONCUR '90 Theories of Concurrency: Unification and Extension* (LNCS 458), pages 263–277, Springer-Verlag, August 1990.

[LS89] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Proceedings of 16th Annual ACM Symposium on Principles of Programming Languages*, 1989.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Volume 92 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency. International Series in Computer Science*, Prentice Hall, 1989.

[Par81] D. M. R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th G.I. Conference on Theoretical Computer Science*, pages 167–183, Springer-Verlag, 1981.

[Plo81] G. D. Plotkin. *A Structural Approach to Operational Semantics.* Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pnu86] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In J. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency*, Springer-Verlag, Berlin, 1986.

[Tof90] C. M. N. Tofts. A synchronous calculus of relative frequency. In J. C. M. Baeten and J. W. Klop, editors, *Proceedings of CONCUR '90*, pages 467–480, Springer-Verlag, Berlin, 1990.

[vGSST90] R. J. van Glabbeek, S. A. Smolka, B. Steffen, and C. M. N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, pages 130–141, Philadelphia, PA, 1990.