A Process Algebra with Distributed Priorities

Rance Cleaveland[1] Gerald Lüttgen[2] V. Natarajan[1]

[1] Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA, e-mail: {rance,nvaidhy}@eos.ncsu.edu . Research supported by NSF grant CCR-9120995, ONR Young Investigator Award N00014-92-J-1582, NSF Young Investigator Award CCR-9257963, NSF grant CCR-9402807, and AFOSR grant F49620-95-1-0508.

[2] Fakultät für Mathematik und Informatik, Universität Passau, 94030 Passau, Germany, e-mail: luettgen@fmi.uni-passau.de . Research support partly provided by the German Academic Exchange Service under grant D/95/09026 (Doktorandenstipendium HSP II/ AUFE).

Abstract. This paper presents a process algebra for distributed systems in which some actions may take precedence over others. In contrast with existing approaches to priorities, our algebra only allows actions to preempt others at the same "location" and therefore captures a notion of *localized precedence*. Using Park's and Milner's notion of strong bisimulation as a basis, we develop a behavioral congruence and axiomatize it for finite processes; we also derive an associated observational congruence. Simple examples highlight the utility of the theory.

1 Introduction

Process algebras [11, 13] provide widely studied frameworks for modeling and verifying concurrent systems [9]. Such theories typically consist of a simple language with a well-defined operational semantics given in terms of labeled transition systems; a behavioral equivalence is then used to relate implementations and specifications, which are both given as terms in the language. In order to facilitate compositional reasoning, in which systems are verified on the basis of the behavior of their components, researchers have devoted great attention to the definition of behavioral congruences, which allow the substitution of "equals for equals" inside larger systems. Traditional process algebras focus on modeling the potential nondeterminism that concurrent processes may exhibit; approaches have also been suggested for introducing sensitivity to other aspects of system behavior, including *priority* [1, 2, 4, 5, 6, 12, 15] and *true concurrency* [3, 14]. The latter work presents theories in which parallelism is treated as a primitive notion that is not reducible to nondeterminism, while the former enables the modeling of systems in which some system transitions (e.g. interrupts) may take precedence over others.

In this paper, we develop an algebraic theory of action priority for *distributed* systems. As in existing work, our aim is to model systems in which some transitions have precedence over others. Our point of departure is that the priority scheme should be localized within individual sites in the system; actions should only be able to pre-empt actions being performed at the "same location." This constraint reflects an essential intuition about distributed systems, which is that the execution of a process on one processor should not affect the behavior of a process on another processor unless the designer explicitly builds in an interaction (e.g. synchronization) between them. Technically, we begin with a theory of priority that includes a notion of global precedence [6, 15] and show how its semantics may be altered using ideas from true concurrency [3] to localize capabilities for pre-emption. We then define a strong congruence for this language, axiomatize it for finite processes, and derive an observational congruence along the lines of [13].

Organization of the Paper. In the next section we present a generic example illustrating the need for local pre-emption in modeling systems. The following three sections present our language and derive the technical results discussed above, while Sect. 6 presents an example showing the application of our theory. Sect. 7 discusses related work, and the last section presents our conclusions and directions for future work. Due to space constraints we refer the reader to [7] for the proofs of our main theorems.

2 Motivating Example

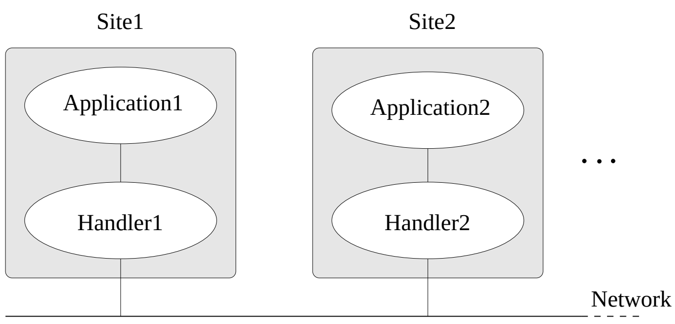

Fig. 1. Standard distributed system

The example depicted in Fig. 1 motivates the need for considering a local notion of pre-emption when dealing with priorities in distributed systems. It consists of two sites, `Site1` and `Site2`, e.g. two computers, that are connected via the network `Network`. Each site runs an application, `Application1` and `Application2`, respectively, which may send or receive information from the application at the other site via its (interrupt-)handler, `Handler1` or `Handler2`. A handler delivers the message to the network or receives a message for its site from the network and notifies the application by sending an interrupt. Now, we have the following intuitive requirements which the semantics of our language $\mathsf{CCS}^{\mathsf{prio}}$ should satisfy in order to reflect the behavior of the system correctly. First, an interrupt of a handler should pre-empt the normal work of the application at its site, i.e. the application should immediately respond to an interrupt request. Second, both sites should be able to perform internal computations that are local to their site without interference from the other site. In particular, internal activities of `Handler1` should not pre-empt those of `Handler2`, and vice versa. While traditional process-algebraic treatments [6, 15] of priority satisfy the first requirement, they typically violate the second, since they allow `Application1` to pre-empt `Application2` if the former has higher priority, even though they are running on different sites. In general, one would expect priorities at different sites to be incomparable. The semantics given in [6, 15], however, do not permit this distinction to be made; the net effect is that some computations that one would expect to find in a distributed system are improperly suppressed. We propose to remedy this shortcoming in this paper by introducing a notion of *local pre-emption*.

3 Syntax and Semantics of $\mathsf{CCS}^{\mathsf{prio}}$

In this section we define the syntax and semantics of our language $\mathsf{CCS}^{\mathsf{prio}}$, which is based on CCS [13].

3.1 Syntax of $\mathsf{CCS}^{\mathsf{prio}}$

The syntax of $\mathsf{CCS}^{\mathsf{prio}}$ differs from CCS in the structure of the action set which exhibits a priority scheme. For the sake of simplicity, we restrict ourselves to a two-level priority scheme. However, all results presented in this paper can be generalized to multi-level priority schemes in a straightforward fashion. Intuitively, actions represent potential synchronizations that a process may be willing to engage in with its environment. Given a choice between a synchronization on a high priority action and one on a low priority action, a process should choose the former.

Formally, let Λ be a countable set of action labels, not including the *internal* or *silent* action τ. For every *input action* $a \in \Lambda$, there exists a *complementary action* $\overline{a}$, the corresponding *output action*. Let $\overline{\Lambda} =_{\mathrm{df}} \{\overline{a} \,|\, a \in \Lambda\}$, and let us denote the set of all actions $\Lambda \cup \overline{\Lambda} \cup \{\tau\}$, where $\tau \notin \Lambda$, by A. Intuitively, an action indicates that a process is willing to perform a synchronization on the *port* associated with the action name, i.e. action a means that the process wants to receive a message from port a whereas $\overline{a}$ means that the process wants to send a message via port a. The action τ represents either an internal action of a process or the synchronization of two processes on some port in order to communicate with each other. Finally, we let $a, b, \ldots$ range over Λ and $\alpha, \beta, \ldots$ over A.

In order to define *prioritized actions*, let $\underline{\Lambda}$ be a countable set of prioritized action labels disjoint from Λ. Then $\underline{A} =_{\mathrm{df}} \underline{\Lambda} \cup \overline{\underline{\Lambda}} \cup \{\underline{\tau}\}$ is the set of *prioritized actions*, where $\underline{\tau}$ is the prioritized *internal* or *silent* action. We use $\mathcal{A} =_{\mathrm{df}} A \cup \underline{A}$

to denote the set of all actions. Intuitively, prioritized actions are considered to be names for "important" channels. Therefore, communications on a prioritized action should be preferred over communications on unprioritized actions. In the remainder of the paper, let $\underline{a}, \underline{b}, \ldots$ range over $\underline{\Lambda}$, the symbols $\underline{\alpha}, \underline{\beta}, \ldots$ over $\underline{A}$, and γ, δ over $\mathcal{A}$. Additionally, we extend $\bar{\ }$ by $\overline{\overline{\gamma}} = \gamma$, and if $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ then $\overline{L} =_{\mathrm{df}} \{\overline{\gamma} \,|\, \gamma \in L\}$. A mapping f on $\mathcal{A}$ is a *relabeling* if f preserves priorities (i.e. $f(\Lambda) \subseteq \Lambda$ and $f(\underline{\Lambda}) \subseteq \underline{\Lambda}$), is such that the set $\{\gamma \,|\, f(\gamma) \neq \gamma\}$ is finite, and satisfies the following: $f(\overline{a}) = \overline{f(a)}$ $f(\overline{\underline{a}}) = \overline{f(\underline{a})}$, $f(\tau) = \tau$, and $f(\underline{\tau}) = \underline{\tau}$.

The syntax of our language is defined by the BNF

$$P \quad ::= \quad \mathbf{0} \mid \gamma.P \mid P + P \mid P|P \mid P[f] \mid P \setminus L \mid C \stackrel{\mathrm{def}}{=} P$$

where f is a relabeling, $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$, and C is a process constant. We use the standard definitions for *sort* of a process, *free* and *bound variables*, *open* and *closed terms*, *guarded recursion*, and *contexts*. We refer to closed and guarded terms as *processes* and denote syntactic equality by $\equiv$. Let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of processes.

3.2 Locations

We now introduce the notion of *location*, which will be used in the next section in the operational semantics for $\mathsf{CCS}^{\mathsf{prio}}$ as a basis for deciding when one transition pre-empts another. Intuitively, a location is a string representing the "address" of a subterm inside a larger term; when a system performs an action, our semantics will also note the location of the subterm that "generates" this action. Our account of locations closely follows that of [14].

Formally, let $\mathcal{A}_{\mathrm{loc}} =_{\mathrm{df}} \{L, R, l, r\}$ be the *location alphabet*, and let $\mathcal{L}oc$ denote the set of all words over $\mathcal{A}_{\mathrm{loc}}$ concatenated with the special symbol $\bullet$ to the left, i.e. $\mathcal{L}oc$ is the set of all *locations*. As usual, $\cdot$ denotes the concatenation operator as e.g. in $\bullet \cdot L \cdot l \in \mathcal{L}oc$. Further, we write $M \cdot \zeta$ for $\{m \cdot \zeta \,|\, m \in M\}$ where $M \subseteq \mathcal{L}oc$ and $\zeta \in \mathcal{A}_{\mathrm{loc}}$. As noted above, a location represents the address of a subterm, with $\bullet$ denoting the current term, l (r) representing the left (right) subterm of a $+$, and L (R) the left (right) component of a $|$. For example, the process $(a.\mathbf{0} \,|\, b.\mathbf{0}) + c.\mathbf{0}$ can perform action a from location $\bullet \cdot L \cdot l$, action b from location $\bullet \cdot R \cdot l$, and action c from location $\bullet \cdot r$. For simplicity, we often write m instead of $\bullet \cdot m$ for $m \in \mathcal{L}oc$.

As mentioned in the introduction, we want to adopt the view that processes on different sides of the parallel operator are (logically) executed on different processors, i.e. at different locations. Thus, priorities on different sides of the parallel operator are distributed and, therefore, should be incomparable. However, processes on different sides of the summation operator, which models non-deterministic choice, are scheduled on a single processor, i.e. they should be comparable. We formalize this intuition in the following *comparability relation* on locations which is adapted from [10].

Definition 1. The comparability relation $\bowtie$ on locations is the smallest reflexive and symmetric subset of $\mathcal{L}oc \times \mathcal{L}oc$ such that for all $v, w \in \mathcal{L}oc$.

1. $(v \cdot l, w \cdot r) \in \bowtie$, and
2. $(v, w) \in \bowtie$ implies $(v \cdot \zeta, w \cdot \zeta) \in \bowtie$ for $\zeta \in \mathcal{A}_{\mathrm{loc}}$.

We write $v \bowtie w$ instead of $(v, w) \in \bowtie$.

Note that the comparability relation is not transitive, e.g. we have $L \cdot l \bowtie r$ and $r \bowtie R \cdot l$ but $L \cdot l \not\bowtie R \cdot l$ since $L \not\bowtie R$. Considering our example $(a.\mathbf{0} \mid b.\mathbf{0}) + c.\mathbf{0}$ above, the locations of the actions a and c and the locations of the actions b and c are comparable since they are just on different sides of the summation operator. In contrast, the locations of the actions a and b are incomparable since they are on different sides of the parallel operator.

In the following, let $m, n, o, \ldots$ range over the set $\mathcal{L}oc$. We let $[m]$ denote the set $\{o \in \mathcal{L}oc \mid o \bowtie m\}$. Moreover, we close the set $\mathcal{L}oc$ with respect to pairing; that is, if $m, n \in \mathcal{L}oc$ then we let $\langle m, n\rangle \in \mathcal{L}oc$ also. Allowing pairs of locations is necessary because communications in a CCS-based framework take place between two processes offering complementary actions. The result of a communication is an internal action which is assigned with the two locations of the complementary actions. Finally, we define $\langle m, n\rangle \cdot \zeta =_{\mathrm{df}} \langle m \cdot \zeta, n \cdot \zeta\rangle$ and $[\langle m, n\rangle] =_{\mathrm{df}} [m] \cup [n]$ where $m, n \in \mathcal{L}oc$ and $\zeta \in \mathcal{A}_{\mathrm{loc}}$.

3.3 Semantics of $\mathsf{CCS}^{\mathsf{prio}}$

The (*operational*) *semantics* of a $\mathsf{CCS}^{\mathsf{prio}}$ process $P \in \mathcal{P}$ is given by a labeled transition system $\langle \mathcal{P}, \mathcal{A}, \longrightarrow, P\rangle$ where $\mathcal{P}$ is the set of states, $\mathcal{A}$ the alphabet, $\longrightarrow$ the transition relation, and P the start state. The transition relation $\longrightarrow \subseteq \mathcal{P} \times (\mathcal{L}oc \times \mathcal{A}) \times \mathcal{P}$ is defined in Table 2 using Plotkin-style operational rules. We write $P \xrightarrow{m,\gamma} P'$ instead of $\langle P, m, \gamma, P'\rangle \in \longrightarrow$. We say that P *may engage in action* γ *offered from location* m *and thereafter behaves like process* P'. Moreover, if $\gamma \in \underline{A}$ then we abbreviate $P \xrightarrow{m,\gamma} P'$ by $P \xrightarrow{\gamma} P'$ since it turns out that the location m is not important when reasoning about prioritized transitions, i.e. transitions labeled by a prioritized action.

The presentation of the operational rules requires *prioritized initial action sets* which are defined as the least relations satisfying the rules in Table 1. Intuitively, $\underline{\mathbb{I}}_m(P)$ denotes the set of all prioritized initial actions of P from location m. Note that those sets are either empty or contain exactly one initial transition. $\underline{\mathbb{I}}_m(P) = \emptyset$ means that either m is not a location of P or P wants to perform an unprioritized action at location m. Additionally, let us denote the set of all prioritized initial actions of process P from locations $M \subseteq \mathcal{L}oc$ by $\underline{\mathbb{I}}_M(P)$, and the set of all prioritized initial actions of process P by $\underline{\mathbb{I}}(P)$. We also define analogous initial action sets ignoring internal actions and denote them by $\underline{\mathbb{II}}_m(P)$, $\underline{\mathbb{II}}_M(P)$, and $\underline{\mathbb{II}}(P)$, respectively.

Note that the set of actions is defined independently from the transition relation $\longrightarrow$. Therefore, $\longrightarrow$ is well-defined. The side conditions of the operational semantic rules guarantee that a process does not perform an unprioritized action if it can engage in a prioritized synchronization or internal computation, i.e. a $\underline{\tau}$-transition, from a comparable location. Therefore, $\underline{\tau}$-actions have pre-emptive

Table 1. Initial action sets

$\underline{\mathrm{I}}_m(C) =_{\mathrm{df}} \underline{\mathrm{I}}_m(P)$ where $C \stackrel{\mathrm{def}}{=} P$	$\underline{\mathrm{I}}_\bullet(\underline{\alpha}.P) =_{\mathrm{df}} \{\underline{\alpha}\}$
$\underline{\mathrm{I}}_{m \cdot l}(P + Q) =_{\mathrm{df}} \underline{\mathrm{I}}_m(P)$	$\underline{\mathrm{I}}_{n \cdot r}(P + Q) =_{\mathrm{df}} \underline{\mathrm{I}}_n(Q)$
$\underline{\mathrm{I}}_m(P[f]) =_{\mathrm{df}} \{f(\underline{\alpha}) \mid \underline{\alpha} \in \underline{\mathrm{I}}_m(P)\}$	$\underline{\mathrm{I}}_m(P \setminus L) =_{\mathrm{df}} \underline{\mathrm{I}}_m(P) \setminus (L \cup \overline{L})$
$\underline{\mathrm{I}}_{m \cdot L}(P\|Q) =_{\mathrm{df}} \underline{\mathrm{I}}_m(P)$	$\underline{\mathrm{I}}_{n \cdot R}(P\|Q) =_{\mathrm{df}} \underline{\mathrm{I}}_n(Q)$
$\underline{\mathrm{I}}_{\langle m \cdot L, n \cdot R \rangle}(P\|Q) =_{\mathrm{df}} \underline{\mathrm{I}}_m(P) \cup \underline{\mathrm{I}}_n(Q) \cup \{\underline{\tau} \mid \underline{\mathrm{II}}_m(P) \cap \overline{\underline{\mathrm{II}}}_n(Q) \neq \emptyset\}$	
$\underline{\mathrm{I}}_M(P) =_{\mathrm{df}} \bigcup\{\underline{\mathrm{I}}_m(P) \mid m \in M\}$	$\underline{\mathrm{II}}_M(P) =_{\mathrm{df}} \underline{\mathrm{I}}_M(P) \setminus \{\underline{\tau}\}$
$\underline{\mathrm{I}}(P) =_{\mathrm{df}} \underline{\mathrm{I}}_{\mathcal{L}oc}(P)$	$\underline{\mathrm{II}}(P) =_{\mathrm{df}} \underline{\mathrm{I}}(P) \setminus \{\underline{\tau}\}$

power over unprioritized actions. The reason that prioritized visible actions do *not* have priority over unprioritized actions is that visible actions only indicate the potential of a synchronization, i.e. the potential of progress, whereas internal actions describe *real* progress in our model.

The semantics of $\mathsf{CCS}^{\mathsf{prio}}$ for prioritized transitions is the same as the usual CCS semantics. The difference arises by the side conditions of the rules for unprioritized transitions. The process $\gamma.P$ may engage in action γ and then behaves like P. The *summation operator* $+$ denotes *nondeterministic choice*. The process $P + Q$ may behave like process P (Q) if Q (P) does not pre-empt unprioritized actions by performing a $\underline{\tau}$-action. Note that priorities arising from different sides of the summation operator are comparable. The *restriction operator* $\setminus L$ prohibits the execution of actions in $L \cup \overline{L}$. Thus, the restriction operator permits the *scoping* of actions. $P[f]$ behaves exactly as the process P where the actions are renamed with respect to the relabeling f. The process $P|Q$ stands for the *parallel composition* of P and Q according to an *interleaving semantics* with *synchronized communication* on complementary actions resulting in the internal action τ or $\underline{\tau}$. Since locations on different sides of a parallel operator are incomparable, $\underline{\tau}$'s arising from a location of P (Q) cannot pre-empt the execution of an action, even an unprioritized one, of Q (P). Only if P (Q) engages in a prioritized synchronization with Q (P) can unprioritized actions of P and Q be pre-empted. Finally, $C \stackrel{\mathrm{def}}{=} P$ denotes a *constant definition*, i.e. C is a recursively defined process which behaves as a distinguished solution of the equation $C = P$.

Table 2. Operational semantics for $\mathsf{CCS}^{\mathsf{prio}}$

<u>Act</u>	$\frac{-}{\underline{\alpha}.P \xrightarrow{\underline{\alpha}} P}$	Act	$\frac{-}{\alpha.P \xrightarrow{\bullet,\alpha} P}$
<u>Sum1</u>	$\frac{P \xrightarrow{\underline{\alpha}} P'}{P+Q \xrightarrow{\underline{\alpha}} P'}$	Sum1	$\frac{P \xrightarrow{m,\alpha} P'}{P+Q \xrightarrow{m\cdot l,\alpha} P'}\ \underline{\tau} \notin \underline{\mathrm{I}}(Q)$
<u>Sum2</u>	$\frac{Q \xrightarrow{\underline{\alpha}} Q'}{P+Q \xrightarrow{\underline{\alpha}} Q'}$	Sum2	$\frac{Q \xrightarrow{n,\alpha} Q'}{P+Q \xrightarrow{n\cdot r,\alpha} Q'}\ \underline{\tau} \notin \underline{\mathrm{I}}(P)$
<u>Rel</u>	$\frac{P \xrightarrow{\underline{\alpha}} P'}{P[f] \xrightarrow{f(\underline{\alpha})} P'[f]}$	Rel	$\frac{P \xrightarrow{m,\alpha} P'}{P[f] \xrightarrow{m,f(\alpha)} P'[f]}$
<u>Res</u>	$\frac{P \xrightarrow{\underline{\alpha}} P'}{P \setminus L \xrightarrow{\underline{\alpha}} P' \setminus L}\ \underline{\alpha} \notin L \cup \overline{L}$	Res	$\frac{P \xrightarrow{m,\alpha} P'}{P \setminus L \xrightarrow{m,\alpha} P' \setminus L}\ \alpha \notin L \cup \overline{L}$
<u>Com1</u>	$\frac{P \xrightarrow{\underline{\alpha}} P'}{P\|Q \xrightarrow{\underline{\alpha}} P'\|Q}$	Com1	$\frac{P \xrightarrow{m,\alpha} P'}{P\|Q \xrightarrow{m\cdot L,\alpha} P'\|Q}\ \underline{\mathbb{I}}_{[m]}(P) \cap \overline{\underline{\mathbb{I}}}(Q) = \emptyset$
<u>Com2</u>	$\frac{Q \xrightarrow{\underline{\alpha}} Q'}{P\|Q \xrightarrow{\underline{\alpha}} P\|Q'}$	Com2	$\frac{Q \xrightarrow{n,\alpha} Q'}{P\|Q \xrightarrow{n\cdot R,\alpha} P\|Q'}\ \underline{\mathbb{I}}_{[n]}(Q) \cap \overline{\underline{\mathbb{I}}}(P) = \emptyset$
<u>Com3</u>	$\frac{P \xrightarrow{\underline{a}} P' \quad Q \xrightarrow{\overline{\underline{a}}} Q'}{P\|Q \xrightarrow{\underline{\tau}} P'\|Q'}$	Com3	$\frac{P \xrightarrow{m,a} P' \quad Q \xrightarrow{n,\overline{a}} Q'}{P\|Q \xrightarrow{\langle m\cdot L, n\cdot R\rangle,\tau} P'\|Q'}$ $\underline{\mathbb{I}}_{[m]}(P) \cap \overline{\underline{\mathbb{I}}}(Q) = \emptyset$ and $\underline{\mathbb{I}}_{[n]}(Q) \cap \overline{\underline{\mathbb{I}}}(P) = \emptyset$
<u>Con</u>	$\frac{P \xrightarrow{\underline{\alpha}} P'}{C \xrightarrow{\underline{\alpha}} P'}\ C \stackrel{\text{def}}{=} P$	Con	$\frac{P \xrightarrow{m,\alpha} P'}{C \xrightarrow{m,\alpha} P'}\ C \stackrel{\text{def}}{=} P$

4 Prioritized Strong Bisimulation

In this section we present an equivalence relation for $\mathsf{CCS}^{\mathsf{prio}}$ processes that is based on bisimulation [17]. Our aim is to characterize the largest congruence contained in the "naive" adaption of strong bisimulation [13] to our framework.

Definition 2 Naive Prioritized Strong Bisimulation. A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called *naive prioritized strong bisimulation* if for every $\langle P, Q\rangle \in \mathcal{R}$, $\gamma \in \mathcal{A}$, and $m \in \mathcal{L}oc$ the following condition holds.

$$P \xrightarrow{m,\gamma} P' \text{ implies } \exists Q', n.\, Q \xrightarrow{n,\gamma} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R}\ .$$

We write $P \simeq Q$ if there exists a naive prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

It is straightforward to establish that $\simeq$ is the *largest* naive prioritized strong bisimulation and that $\simeq$ is an equivalence relation. Unfortunately, $\simeq$ is *not* a congruence, which is a necessary requirement for an equivalence to be suitable for compositional reasoning. The lack of compositionality is demonstrated by the following example, which presents the traditional view of process algebras that "parallelism = nondeterminism." We have $a.\underline{b}.\mathbf{0} + \underline{b}.a.\mathbf{0} \simeq a.\mathbf{0} \,|\, \underline{b}.\mathbf{0}$ but $(a.\underline{b}.\mathbf{0} + \underline{b}.a.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0} \not\simeq (a.\mathbf{0} \,|\, \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$ since the latter can perform an a-transition while the corresponding a-transition of the former process is pre-empted because the right process in the summation can engage in a $\underline{\tau}$-transition.

The above observation is not surprising since the distribution of processes influences the pre-emption of transitions and, consequently, the bisimulation. Thus, in order to find the largest congruence relation $\simeq^+$ contained in $\simeq$ we have to take the *local* pre-emption of processes into account. In the following, we define *prioritized strong bisimulation* $\simeq^+$, which is indeed the largest congruence contained in $\simeq$.

Definition 3 Prioritized Strong Bisimulation. A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *prioritized strong bisimulation* if for every $\langle P, Q \rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions hold.

1. $P \overset{\underline{\alpha}}{\Rightarrow} P'$ implies $\exists Q'.\, Q \overset{\underline{\alpha}}{\Rightarrow} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
2. $P \xrightarrow{m,\alpha} P'$ implies $\exists Q', n.\, Q \xrightarrow{n,\alpha} Q'$, $\underline{\mathbb{I}}_{[n]}(Q) \subseteq \underline{\mathbb{I}}_{[m]}(P)$, and $\langle P', Q' \rangle \in \mathcal{R}$.

We write $P \simeq^+ Q$ if there exists a prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q \rangle \in \mathcal{R}$.

The difference between this definition and the definition of $\simeq$ is the additional requirement concerning the initial action sets, parameterized with the appropriate locations, in the condition for unprioritized transitions. Intuitively, the prioritized initial action set of a process with respect to some location, and not the location itself, is a measure of the pre-emptive power of the process relative to that location. Thus, the second condition of Definition 3 states that an unprioritized action α from some location m of the process P has to be matched by the same action from some location n of Q and that the pre-emptive power of Q with respect to n is at most as strong as the pre-emptive power of P with respect to m.

Proposition 4. *The relation $\simeq^+$ is a congruence, i.e. for all* $\mathsf{CCS}^{\mathsf{prio}}$ *contexts $C[X]$ we have: $P \simeq^+ Q$ implies $C[P] \simeq^+ C[Q]$.*

Theorem 5. *The congruence $\simeq^+$ is the* largest *congruence contained in $\simeq$.*

Axiomatization of $\simeq^+$

In this section we give an axiomatization of $\simeq^+$ for *finite* processes, i.e. processes that do not contain recursion. In order to develop the axiomatization, we need to add a new, binary summation operator $\oplus$ to the process algebra $\mathsf{CCS}^{\mathsf{prio}}$. This operator is called *distributed summation* and needed for giving an expansion axiom for $|$. Its semantics is similar to $+$ except that priorities on different sides of the operator are considered as incomparable.

Definition 6 Distributed Summation. The semantics of the new binary operator $\oplus$ on processes is defined by the following operational rules.

$$\underline{\text{iSum1}}\ \frac{P \stackrel{\underline{\alpha}}{\longrightarrow} P'}{P \oplus Q \stackrel{\underline{\alpha}}{\longrightarrow} P'} \qquad \text{iSum1}\ \frac{P \xrightarrow{m,\alpha} P'}{P \oplus Q \xrightarrow{m\cdot L,\alpha} P'}$$

$$\underline{\text{iSum2}}\ \frac{Q \stackrel{\underline{\alpha}}{\longrightarrow} Q'}{P \oplus Q \stackrel{\underline{\alpha}}{\longrightarrow} Q'} \qquad \text{iSum2}\ \frac{Q \xrightarrow{n,\alpha} Q'}{P \oplus Q \xrightarrow{n\cdot R,\alpha} Q'}$$

Table 3. Axiomatization of $\simeq^+$ (Part I)

(A1)	$x + y = y + x$	(iA1)	$x \oplus y = y \oplus x$
(A2)	$x + (y + z) = (x + y) + z$	(iA2)	$x \oplus (y \oplus z) = (x \oplus y) \oplus z$
(A3)	$x + x = x$	(iA3)	$x \oplus x = x$
(A4)	$x + \mathbf{0} = x$	(iA4)	$x \oplus \mathbf{0} = x$

(P) $\quad \underline{\tau}.x + \alpha.y = \underline{\tau}.x$

(E) $\quad P \equiv \bigoplus_i \sum_j \gamma_{ij}.P_{ij}$ and $Q \equiv \bigoplus_k \sum_l \delta_{kl}.Q_{kl}$ implies

$$\begin{aligned}
&P\,|\,Q = \\
&\textstyle\bigoplus_i \sum_j (\gamma_{ij}.(P_{ij}\,|\,Q) + \sum_k \sum_l \{\tau.(P_{ij}\,|\,Q_{kl}) \mid \gamma_{ij} = \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad\qquad \textstyle + \sum_k \sum_l \{\underline{\tau}.(P_{ij}\,|\,Q_{kl}) \mid \gamma_{ij} = \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\}) \oplus \\
&\textstyle\bigoplus_k \sum_l (\delta_{kl}.(P\,|\,Q_{kl}) + \sum_i \sum_j \{\tau.(P_{ij}\,|\,Q_{kl}) \mid \gamma_{ij} = \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad\qquad \textstyle + \sum_i \sum_j \{\underline{\tau}.(P_{ij}\,|\,Q_{kl}) \mid \gamma_{ij} = \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\})
\end{aligned}$$

(Res1)	$\mathbf{0} \setminus L = \mathbf{0}$		(Rel1)	$\mathbf{0}[f] = \mathbf{0}$
(Res2)	$(\gamma.x) \setminus L = \mathbf{0}$	$(\gamma \in L \cup \overline{L})$	(Rel2)	$(\gamma.x)[f] = f(\gamma).(x[f])$
(Res3)	$(\gamma.x) \setminus L = \gamma.(x \setminus L)$	$(\gamma \notin L \cup \overline{L})$	(Rel3)	$(x + y)[f] = x[f] + y[f]$
(Res4)	$(x + y) \setminus L = (x \setminus L) + (y \setminus L)$		(iRel3)	$(x \oplus y)[f] = x[f] \oplus y[f]$
(iRes4)	$(x \oplus y) \setminus L = (x \setminus L) \oplus (y \setminus L)$			

Now, we turn to the axiom system for prioritized strong bisimulation. We write $\vdash P = Q$ if P can be rewritten to Q using the axioms in the Tables 3 and 4. Axioms (S2) and (S3) involve side conditions. The relation $\sqsubseteq_{\mathrm{i}}$ is the precongruence on finite processes generated from the axioms (iC1), (iC2), and (iC3) using the laws of inequational reasoning. The axioms in Table 3 are basically those presented in [6] augmented with the corresponding axioms for the incomparable summation operator. Moreover, the expansion axiom has been adapted for our

Table 4. Axiomatization of $\simeq^+$ (Part II) and axiomatization of $\sqsubseteq_{\mathrm{i}}$

(D1) $(x \oplus \alpha.y) + \beta.z = (x + \beta.z) \oplus \alpha.y$	(iC1) $\underline{\alpha}.x \sqsubseteq_{\mathrm{i}} \underline{\alpha}.y$	
(D2) $(x \oplus \underline{\alpha}.y) + \beta.z = x \oplus (x + \underline{\alpha}.y + \beta.z)$	(iC2) $\mathbf{0} \sqsubseteq_{\mathrm{i}} \nu.x \quad (\nu \in \mathcal{A} \setminus \{\underline{\tau}\})$	
(D3) $(x \oplus y) + \underline{\alpha}.z = (x + \underline{\alpha}.z) \oplus (y + \underline{\alpha}.z)$	(iC3) $\alpha.x \sqsubseteq_{\mathrm{i}} \mathbf{0}$	
(Ic1) $\underline{\alpha}.x + \underline{\beta}.y = \underline{\alpha}.x \oplus \underline{\beta}.y$		
(Ic2) $(\underline{\alpha}.x + y) = (\underline{\alpha}.x + y) \oplus \underline{\alpha}.x$		
(S1) $(x + \underline{\alpha}.y) \oplus (x' + \underline{\alpha}.y') = (x + \underline{\alpha}.y + \underline{\alpha}.y') \oplus (x' + \underline{\alpha}.y + \underline{\alpha}.y')$		
(S2) $(x + \alpha.z) \oplus (y + \alpha.z) = (x + \alpha.z) \oplus y$		$(\vdash x \sqsubseteq_{\mathrm{i}} y)$
(S3) $x \oplus y = x + y$		$(\vdash x =_{\mathrm{i}} y)$

algebra (cf. Axiom (E) where $\sum$ is the indexed version of $+$ and $\bigoplus$ the indexed version of the new summation operator $\oplus$). The axioms in Table 4 are new and show how we may "restructure" locations. They deal with the *distributivity* of the summation operators (Axioms (D1), (D2), and (D3)), the *interchangeability* of the summation operators (Axioms (Ic1) and (Ic2)), and the *saturation* of locations (Axioms (S1), (S2), and (S3)), respectively.

Lemma 7. *Let* $\vdash P \sqsubseteq_{\mathrm{i}} Q$ *for some processes* $P, Q \in \mathcal{P}$. *Then,* $\underline{\mathbb{I}}(P) \subseteq \underline{\mathbb{I}}(Q)$ *holds. Moreover,* $\underline{\tau} \in \underline{\mathrm{I}}(P)$ *if and only if* $\underline{\tau} \in \underline{\mathrm{I}}(Q)$.

We write $\vdash P =_{\mathrm{i}} Q$ iff $\vdash P \sqsubseteq_{\mathrm{i}} Q$ and $\vdash Q \sqsubseteq_{\mathrm{i}} P$. Considering the meaning of the side conditions as made precise in Lemma 7, it is immediately clear that the Axioms (S2) and (S3) are sound. In order to prove our axiomatization complete, we introduce a notion of *normal form* of processes that is based on the following definition.

Definition 8 Summation Form. A process $P \in \mathcal{P}$ is in *summation form* if it has the form $P \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.P_{ij}$ where $m, n_i \in \mathbb{N}$ and the processes P_{ij} are again in summation form. Per definition, $\mathbf{0}$ is in summation form.

Intuitively, P is distributed throughout m incomparable locations which themselves consist of n_i comparable locations, $1 \leq i \leq m$. Now, we are able to define *normal forms*.

Definition 9 Normal Form. Let $P \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.P_{ij}$ be in summation form. We define $\underline{\gamma}_{i*} =_{\mathrm{df}} \{\gamma_{ij} \mid 1 \leq j \leq n_i\} \cap \underline{A}$. The process P is said to be in *normal form* if the following properties hold.

1. $\emptyset \subset L \subseteq \underline{\mathrm{I}}(P)$ implies $\exists i.\, \underline{\gamma}_{i*} = L$.
2. $\gamma_{ij} = \underline{\tau}$ and $\gamma_{kl} \in A$ imply $i \neq k$.
3. $\gamma_{ij} = \gamma_{kl} = \underline{\alpha}$ implies $\exists j'.\, P_{ij'} \equiv P_{kl}$ and $\gamma_{ij'} = \underline{\alpha}$.

4. $i \neq k$ implies $\underline{\gamma}_{i*} \neq \underline{\gamma}_{k*}$.
5. $\gamma_{ij}.P_{ij} \equiv \gamma_{kl}.P_{kl}$, $\gamma_{ij} \in A$, and $i \neq k$ imply $\underline{\gamma}_{i*} \not\subset \underline{\gamma}_{j*}$.

Proposition 10. *If P is a finite process, then there exists a normal form N such that $\vdash N = P$.*

Rewriting a process in its normal form requires restructuring its locations. After this is done, standard techniques used in CCS (cf. [13]) can be applied in order to show our axiomatization complete.

Theorem 11 Soundness & Completeness. *For finite processes $P, Q \in \mathcal{P}$ we have $\vdash P = Q$ if and only if $P \simeq^+ Q$.*

5 Prioritized Observational Congruence

The semantical congruence developed in the previous section is too strong for verifying systems in practice, as it requires that two equivalent terms match each other's transitions exactly, even those labeled by internal actions. In this section we remedy this problem by developing a semantic congruence that abstracts away from internal transitions. Our approach follows the lines of [15, 13]. We start off with the definition of a naive prioritized weak bisimulation which abstracts from internal actions. This relation is an adaption of observational equivalence [13].

Definition 12 Naive Weak Transition Relation. We define:

1. $\hat{\gamma} =_{\mathrm{df}} \epsilon$ if $\gamma \in \{\underline{\tau}, \tau\}$ and $\hat{\gamma} =_{\mathrm{df}} \gamma$, otherwise.
2. $\stackrel{\epsilon}{\Longrightarrow}_\times =_{\mathrm{df}} (\stackrel{\underline{\tau}}{\longrightarrow} \cup \{ \stackrel{m,\tau}{\longrightarrow} \mid m \in \mathcal{L}oc\})^*$
3. $\stackrel{\alpha}{\Longrightarrow}_\times =_{\mathrm{df}} \stackrel{\epsilon}{\Longrightarrow}_\times \circ \stackrel{\underline{\alpha}}{\longrightarrow} \circ \stackrel{\epsilon}{\Longrightarrow}_\times$
4. $\stackrel{m,\alpha}{\Longrightarrow}_\times =_{\mathrm{df}} \stackrel{\epsilon}{\Longrightarrow}_\times \circ \stackrel{m,\alpha}{\longrightarrow} \circ \stackrel{\epsilon}{\Longrightarrow}_\times$

Definition 13 Naive Prioritized Weak Bisimulation. A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *naive prioritized weak bisimulation* if for every $\langle P, Q \rangle \in \mathcal{R}$, $\gamma \in \mathcal{A}$, and $m \in \mathcal{L}oc$ the following condition holds.

$$P \stackrel{m,\gamma}{\longrightarrow} P' \text{ implies } \exists Q', n.\, Q \stackrel{n,\hat{\gamma}}{\Longrightarrow}_\times Q' \text{ and } \langle P', Q' \rangle \in \mathcal{R} \ .$$

We write $P \approx_\times Q$ if there exists a naive prioritized weak bisimulation $\mathcal{R}$ such that $\langle P, Q \rangle \in \mathcal{R}$.

It is fairly easy to see that $\approx_\times$ is not a congruence for $\mathsf{CCS}^{\mathsf{prio}}$. On the other hand, it reflects an intuitive approach to abstracting away from internal computation, and consequently we devote the rest of this section to characterizing the largest congruence contained in this relation. To do so, we first redefine the weak transition relation as follows.

Definition 14 Prioritized Weak Transition Relation. For $L, M \subseteq \underline{A} \setminus \{\underline{\tau}\}$ we define the following notations.

1. $\hat{\underline{\tau}} =_{\mathrm{df}} \underline{\epsilon}$, $\hat{\underline{a}} =_{\mathrm{df}} \underline{a}$, $\hat{\tau} =_{\mathrm{df}} \epsilon$, and $\hat{a} =_{\mathrm{df}} a$.
2. $P \overset{m,\alpha}{\underset{L}{\longrightarrow}} P'$ iff $P \xrightarrow{m,\alpha} P'$ and $\underline{\mathbb{I}}_{[m]}(P) \subseteq L$.
3. $\overset{\underline{\epsilon}}{\Longrightarrow} =_{\mathrm{df}} (\overset{\underline{\tau}}{\longrightarrow} \cup \{\overset{m,\tau}{\underset{\emptyset}{\longrightarrow}} \mid m \in \mathcal{L}oc\})^*$
4. $\overset{\underline{\alpha}}{\Longrightarrow} =_{\mathrm{df}} \overset{\underline{\epsilon}}{\Longrightarrow} \circ \overset{\underline{\alpha}}{\longrightarrow} \circ \overset{\underline{\epsilon}}{\Longrightarrow}$
5. $\overset{\epsilon}{\underset{L}{\Longrightarrow}} =_{\mathrm{df}} (\overset{\underline{\tau}}{\longrightarrow} \cup \{\overset{m,\tau}{\underset{L}{\longrightarrow}} \mid m \in \mathcal{L}oc\})^*$
6. $P \overset{m,\alpha}{\underset{L,M}{\Longrightarrow}} P'$ iff $\exists P'', P'''.\, P \overset{\epsilon}{\underset{L}{\Longrightarrow}} P'' \overset{m,\alpha}{\underset{L}{\longrightarrow}} P''' \overset{\underline{\epsilon}}{\Longrightarrow} P'$ and $\underline{\mathbb{I}}(P'') \subseteq M$.

Intuitively, $P \overset{m,\alpha}{\underset{L}{\longrightarrow}} P'$ means that P can evolve to P' by performing action α from location m and the pre-emptive power of P at location m is at most L. Recall that the prioritized initial action set of a process (with respect to a location) is a measure of its pre-emptive power. Actually, there are two slightly different views of pre-emption which are encoded in the sets L and M in the definition of $P \overset{m,\alpha}{\underset{L,M}{\Longrightarrow}} P'$, respectively. Whereas L is concerned with the influence of the environment, i.e. a parallel context, on actions performed on the path from P to P''', the set M reflects the impact of P'' on potential synchronization partners (cf. Rule Com3). Note that the definition of $P \overset{\epsilon}{\underset{L}{\Longrightarrow}} P'$ corresponds with our intuition that internal actions, and, therefore, their locations are unobservable. Additionally, a parallel context of P is not influenced by internal actions performed by P since priorities arising from different sides of the parallel operator are incomparable. Therefore, the parameter M is unnecessary in the definition of the relation $\overset{\epsilon}{\underset{L}{\Longrightarrow}}$.

Definition 15 Prioritized Weak Bisimulation. A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *prioritized weak bisimulation* if for every $\langle P, Q \rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions hold.

1. $\exists Q', Q''.\, Q \overset{\underline{\epsilon}}{\Longrightarrow} Q'' \overset{\underline{\epsilon}}{\Longrightarrow} Q'$, $\underline{\mathbb{I}}(Q'') \subseteq \underline{\mathbb{I}}(P)$, and $\langle P, Q' \rangle \in \mathcal{R}$.
2. $P \overset{\underline{\alpha}}{\longrightarrow} P'$ implies $\exists Q'.\, Q \overset{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
3. $P \xrightarrow{m,\alpha} P'$ implies $\exists Q', n.\, Q \overset{n,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q'$, $L = \underline{\mathbb{I}}_{[m]}(P)$, $M = \underline{\mathbb{I}}(P)$, and $\langle P', Q' \rangle \in \mathcal{R}$.

We write $P \approx\!\!\!\!\approx Q$ if there exists a prioritized weak bisimulation $\mathcal{R}$ such that $\langle P, Q \rangle \in \mathcal{R}$.

From this definition, we may directly conclude that $\approx\!\!\!\!\approx$ is the *largest* prioritized weak bisimulation, and that $\approx\!\!\!\!\approx$ is an equivalence relation. The first condition of Definition 15 guarantees that prioritized weak bisimulation is compositional with respect to the parallel operator. Its necessity is best illustrated by the following example. The processes $P \overset{\mathrm{def}}{=} \underline{\tau}.\underline{a}.\mathbf{0}$ and $Q \overset{\mathrm{def}}{=} \underline{a}.\mathbf{0}$ would be considered as equivalent if the first condition would be absent. However, the context $C[X] \overset{\mathrm{def}}{=} X \,|\, (\overline{\underline{a}}.\mathbf{0} + b.\mathbf{0})$ is able to distinguish them.

Proposition 16. *The equivalence relation* $\underline{\approx}$ *is a congruence with respect to all* $\mathsf{CCS}^{\mathsf{prio}}$ *operators except the summation operator* $+$, *the distributed summation operator* $\oplus$, *and recursion.*

In contrast to [15], the summation fix presented in [13] is not sufficient in order to achieve a congruence relation. E.g., let $C \stackrel{\text{def}}{=} \underline{\tau}.D$ and $D \stackrel{\text{def}}{=} \tau.C$. Now, define $P \stackrel{\text{def}}{=} \tau.C$ and $Q \stackrel{\text{def}}{=} \underline{\tau}.D$. By Definition 15 we may observe $P \,\underline{\approx}\, Q$, but $P + a.\mathbf{0} \not\underline{\approx}\, Q + a.\mathbf{0}$ since the former can perform an a-action whereas the latter cannot. It turns out that we have to require that observationally congruent processes must have the same initial actions. This requirement is stronger than the first condition of Definition 15.

Definition 17 Prioritized Observational Congruence. We define $P \,\underline{\approx}^{+}\, Q$ if for all $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions hold.

1. $\underline{\mathrm{I}}(P) = \underline{\mathrm{I}}(Q)$
2. $P \stackrel{\underline{\alpha}}{\rightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\underline{\alpha}}{\Longrightarrow} Q'$ and $P' \,\underline{\approx}\, Q'$.
3. $P \xrightarrow{m,\alpha} P'$ implies $\exists Q', n.\, Q \underset{L,M}{\stackrel{n,\alpha}{\Longrightarrow}} Q'$, $L = \underline{\mathbb{I}}_{[m]}(P)$, $M = \underline{\mathbb{I}}(P)$, and $P' \,\underline{\approx}\, Q'$.
4. $Q \stackrel{\underline{\alpha}}{\rightarrow} Q'$ implies $\exists P'.\, P \stackrel{\underline{\alpha}}{\Longrightarrow} P'$ and $P' \,\underline{\approx}\, Q'$.
5. $Q \xrightarrow{m,\alpha} Q'$ implies $\exists P', n.\, P \underset{L,M}{\stackrel{n,\alpha}{\Longrightarrow}} P'$, $L = \underline{\mathbb{I}}_{[m]}(Q)$, $M = \underline{\mathbb{I}}(Q)$, and $P' \,\underline{\approx}\, Q'$.

Theorem 18. *The relation* $\underline{\approx}^{+}$ *is the largest congruence contained in* $\approx_{\times}$.

The proof of this theorem makes use of the presence of the distributed summation operator in $\mathsf{CCS}^{\mathsf{prio}}$.

6 Example

In this section we demonstrate the utility of $\mathsf{CCS}^{\mathsf{prio}}$ for the verification of distributed systems using an example involving an architecture scheme found in many of today's computers.

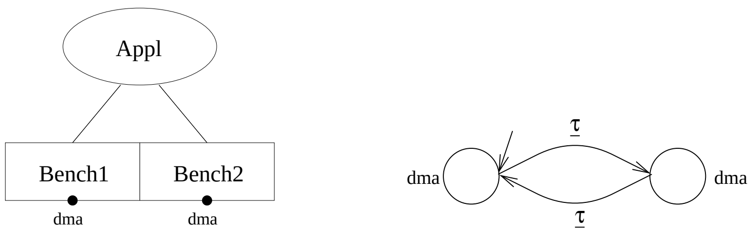

Fig. 2. Example system and its semantics

Our example system consists of an application which receives and writes data from two memory benches (cf. Fig. 2, left hand side). In order to improve the efficiency in a computer system each bench is connected to a direct-memory-access (DMA) controller. To overcome the low speed of most memory modules, the application $\mathtt{Appl}$ works alternately with each memory bench. We model $\mathtt{Appl}$ in $\mathsf{CCS}^{\mathsf{prio}}$ by $\mathtt{Appl} \stackrel{\text{def}}{=} \underline{\mathtt{fetch1}}.\underline{\mathtt{fetch2}}.\mathtt{Appl}$. Each memory bench $\mathtt{Bench1}$ and $\mathtt{Bench2}$ is continuously able to serve the application or to allow the external DMA controller to access the memory via the channel $\mathtt{dma}$. However, if a memory bench has to decide between both activities, then it chooses the former since the progress of the application is considered as more important. Consequently, we define $\mathtt{Bench1} \stackrel{\text{def}}{=} \overline{\underline{\mathtt{fetch1}}}.\mathtt{Bench1} + \mathtt{dma}.\mathtt{Bench1}$ and $\mathtt{Bench2} \stackrel{\text{def}}{=} \overline{\underline{\mathtt{fetch2}}}.\mathtt{Bench2} + \mathtt{dma}.\mathtt{Bench2}$. The overall system $\mathtt{Sys}$ is given by $\mathtt{Sys} \stackrel{\text{def}}{=} (\mathtt{Appl} \,|\, \mathtt{Bench1} \,|\, \mathtt{Bench2}) \setminus \{\mathtt{fetch1}, \mathtt{fetch2}\}$. Since the application uses the memory cells alternately, the DMA is expected to be allowed to access the free memory bench. Therefore, the specification is simply $\mathtt{Spec} \stackrel{\text{def}}{=} \mathtt{dma}.\mathtt{Spec}$. The $\mathsf{CCS}^{\mathsf{prio}}$ semantics of $\mathtt{Sys}$ is given in Fig. 2, right hand side, where we abstract from the locations of the action $\mathtt{dma}$. It is easy to see that the symmetric closure of

$$\{\langle \mathtt{Spec}, \mathtt{Sys} \rangle, \langle \mathtt{Spec}, (\underline{\mathtt{fetch2}}.\mathtt{Appl} \,|\, \mathtt{Bench1} \,|\, \mathtt{Bench2}) \setminus \{\mathtt{fetch1}, \mathtt{fetch2}\} \rangle\}$$

is a prioritized weak bisimulation. Note that Condition (1) of Definition 15 is trivially satisfied since $\mathtt{Spec}$ and $\mathtt{Sys}$ do not contain any visible prioritized actions. Therefore, we obtain $\mathtt{Spec} \underline{\approx} \mathtt{Sys}$ as expected. However, in the traditional approach [6, 15] the $\mathtt{dma}$-loops in the labeled transition system of $\mathtt{Sys}$ would be missing, and $\mathtt{Sys}$ would not be observationally equivalent to $\mathtt{Spec}$.

7 Discussion and Related Work

Several proposals have been made for extending traditional process algebras with priorities. They differ in the aspects of computation, such as interrupts [1], programming constructs like the `PRIALT` construct of `occam` [5, 12], or real-time [4], that they aim to capture.

An extension of CCS [13] with priorities has been proposed in [6], where priorities are assigned to actions in a *globally dynamic* way, i.e. in one state of a system action α may have priority over action β while the situation may be converse in another state of the system. For that process algebra a complete semantic theory has been developed in an analogous fashion to [13] which includes congruences based on strong and weak bisimulation and their axiomatic characterizations [15].

Our process algebra $\mathsf{CCS}^{\mathsf{prio}}$ is based on the approach in [6, 15], where we adopt all design decisions except the notion of *global* pre-emption. Therefore, $\mathsf{CCS}^{\mathsf{prio}}$ has the following characteristics. Only transitions labeled by complementary actions with the same priority may engage in a synchronization. As in [6], we consider actions with different priorities as *different* channels. This is sufficient for most cases occurring in practice [8] and avoids that priorities values have to be adjusted in case of communication (cf. [4, 10]). The strong relation of $\mathsf{CCS}^{\mathsf{prio}}$ to the process algebra proposed in [6, 15] can be made precise by the following fact. If we globalize pre-emption in our framework by defining $[m] =_{\mathrm{df}} \mathcal{L}oc$ for all $m \in \mathcal{L}oc$, our operational semantics and our behavioral relations reduce to the corresponding notions presented in [6, 15].

For a comparison with our work it is of importance that all the above mentioned traditional approaches are provided with a semantics which deals with *global* pre-emption. In contrast, we consider a notion of *local* pre-emption. This idea is also presented in [10], where a CSP-based language is extended with priorities. However, this process algebra suffers from a complicated semantics, especially for the hiding operator. The authors only conjecture that their strong bisimulation is a congruence. They do not provide an axiomatization for their equivalence and do not present a theory for observational congruence. Also Prasad's *Calculus of Broadcasting Systems with Priorities* (PCBS) [18] deals with a distributed notion of priorities. For PCBS a nice semantic theory based on bisimulation has been developed. However, our process algebra $\mathsf{CCS}^{\mathsf{prio}}$ is concerned with a different model for communication.

We close this section with some remarks about our notion of strong and weak bisimulation. Since our semantic theory reflects local pre-emption, locations are implicitly occurring in our semantic equivalences. However, in contrast to [3] locations are not explicitly considered in our bisimulations. Our objective is not to observe locations but to observe local pre-emption which is necessary for causal reasoning in process algebras with priorities.

8 Conclusions and Future Work

In this paper we have presented a process algebra, $\mathsf{CCS}^{\mathsf{prio}}$, with distributed priorities. The key idea for this algebra is to take the distribution of the considered system into account for defining a notion of *local* pre-emption. We have developed a semantic theory for this algebra and have shown its suitability by an example. However, it remains to show how our prioritized bisimulations can be computed before implementing $\mathsf{CCS}^{\mathsf{prio}}$ in an automated verification tool [9]. In order to apply standard algorithms [16] the bisimulations have to be characterized using a transition relation that is not parameterized with prioritized initial action sets. Moreover, we intend to axiomatize prioritized observational congruence.

References

1. 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*, pages 127–168, 1986.
2. E. Best and M. Koutny. Petri net semantics of priority systems. *Theoretical Computer Science*, 96:175–215, 1992.

3. G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114(1):31–61, June 1993.
4. P. Brémonde-Grégoire, I. Lee, and R. Gerber. ACSR: An algebra of communicating shared resources with dense time and priorities. In E. Best, editor, *CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 417–431, Hildesheim, Germany, August 1993. Springer-Verlag.
5. J. Camilleri and G. Winskel. CCS with priority choice. *Information and Computation*, 116(1):26–37, January 1995.
6. R. Cleaveland and M.C.B. Hennessy. Priorities in process algebra. *Information and Computation*, 87(1/2):58–77, July/August 1990.
7. R. Cleaveland, G. Lüttgen, and V. Natarajan. A process algebra with distributed priorities. Technical Report TR-96-02, North Carolina State University, March 1996.
8. R. Cleaveland, G. Lüttgen, V. Natarajan, and S. Sims. Priorities for modeling and verifying distributed systems. In T. Margaria and B. Steffen, editors, *Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 278–297, Passau, Germany, March 1996. Springer-Verlag.
9. R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based tool for the verification of finite-state systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.
10. H. Hansson and F. Orava. A process calculus with incomparable priorities. In *Proceedings of the North American Process Algebra Workshop*, Workshops in Computing, pages 43–64, Stony Brook, New York, August 1992. Springer-Verlag.
11. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, London, 1985.
12. C.-T. Jensen. *Prioritized and Independent Actions in Distributed Computer Systems.* PhD thesis, Aarhus University, August 1994.
13. R. Milner. *Communication and Concurrency.* Prentice-Hall, London, 1989.
14. U. Montanari and D. Yankelevich. A parametric approach to localities. In W. Kuich, editor, *Automata, Languages and Programming (ICALP '92)*, volume 623 of *Lecture Notes in Computer Science*, pages 617–628, Vienna, July 1992. Springer-Verlag.
15. V. Natarajan, L. Christoff, I. Christoff, and R. Cleaveland. Priorities and abstraction in process algebra. In P.S. Thiagarajan, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 880 of *Lecture Notes in Computer Science*, pages 217–230, Madras, India, December 1994. Springer-Verlag.
16. R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.
17. D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th G.I. Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1980.
18. K. V. S. Prasad. Broadcasting with priority. In *Proceedings of the 5th European Symposium on Programming*, volume 788 of *Lecture Notes in Computer Science*, pages 469–484, Edinburgh, U.K., April 1994. Springer-Verlag.

This article was processed using the LaTeX macro package with LLNCS style