Available online at www.sciencedirect.com
ScienceDirect
Information and Computation 205 (2007) 1426–1458
Information and Computation
www.elsevier.com/locate/ic

Priority and abstraction in process algebra

Rance Cleaveland [a], Gerald Lüttgen [b,*], V. Natarajan [c]

[a] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*
[b] *Department of Computer Science, University of York, York YO10 5DD, UK*
[c] *Systems and Technology Group, IBM Corporation, Research Triangle Park, NC 27709, USA*

Received 9 June 2006; revised 16 April 2007
Available online 29 May 2007

Abstract

More than 15 years ago, Cleaveland and Hennessy proposed an extension of the process algebra CCS in which some actions may take priority over others. The theory was equipped with a behavioral congruence based on strong bisimulation.

This article gives a full account of the challenges in, and the solutions employed for, defining a semantic theory of observation congruence for this process algebra. A full-abstraction result is presented whose proof relies on a novel approach based on successive approximations for identifying the largest congruence contained in an intuitive but naïve equivalence. Prioritized observation congruence is also characterized equationally for the class of finite processes, while its utility for system verification is demonstrated by an illustrative example.

Keywords: Process algebra; Priority; Bisimulation; Observation congruence; Full-abstraction; Axiomatization

1. Introduction

Over the past 25 years, *process algebras* [5], such as Milner's Calculus of Communicating Systems (CCS) [25], have been developed for modeling and reasoning about the communication behavior of concurrent systems. Process-algebraic theories focus on behavioral equivalences as a means for verifying systems: one typically formulates a specification as a process describing the desired behavior of a system and then proves that an implementation process is correct by showing that it is equivalent to (or "behaves the same as") the specification. Provided the equivalence is a congruence, one may also reason about the correctness of a system compositionally, on the basis of the correctness of its components.

While traditional process algebras are devoted to modeling potential non-determinism that systems may exhibit during their execution, researchers have also suggested extensions that introduce sensitivity to other aspects of system behavior, including *priority*. This latter work is intended to allow the modeling of different

* Corresponding author. Fax: +44 1904 432767.
E-mail addresses: rance@cs.umd.edu (R. Cleaveland), luettgen@cs.york.ac.uk (G. Lüttgen), nataraj@us.ibm.com (V. Natarajan).

0890-5401/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2007.05.001

priority levels among the actions that processes may perform, so that systems in which some actions, e.g., interrupts, take precedence over others may be modeled faithfully. The first to study priority in process algebra were Baeten et al. [3]; relevant recent work may be found in [1,2,4,8,29]. A survey of results is contained in [13].

1.1. The Cleaveland and Hennessy approach to priority

One well-studied approach to priority in process algebra was introduced by Cleaveland and Hennessy in [10]. It has inspired follow-up research regarding not only priority [9,12,16], but also real time [6,11,20,24] where action priority corresponds to action urgency. Technically, Cleaveland and Hennessy's work extends Milner's CCS [25] by distinguishing between prioritized and unprioritized actions, so that synchronizations on prioritized actions *preempt* synchronizations on unprioritized actions, and by adding prioritization and deprioritization operators for adjusting action priorities. The resulting language, to which we refer as CCS^{prio}, comes equipped with a behavioral congruence based on strong bisimulation [25], which has been completely axiomatized for those finite processes that do not contain any prioritization and deprioritization operators [10]. However, this congruence does not attempt any abstraction from internal computation, which imposes severe restrictions on the use of these equivalences in verification. This is because a user wishing to establish the equivalence of two systems must account for their precise level of internal computation.

1.2. This article

The first contribution of this article is a congruence on CCS^{prio} processes that is as insensitive to internal computation as possible, in a precisely defined sense. Specifically, we start from the usual notion of *observation equivalence* [25] adapted to the setting of CCS^{prio} and observe that compositionality is violated with respect not only to choice (as is standard), but also to parallel composition. The challenge regarding parallel composition is due to system contexts being able to offer synchronizations on prioritized actions and thus to preempt unprioritized actions, and is also a consequence of loops of prioritized internal actions being able to preempt unprioritized actions indefinitely. We then develop a modified observational equivalence, *prioritized observation congruence*, and prove that it is the largest congruence contained inside the original equivalence. In this sense prioritized observation congruence is fully abstract. The proof of this result relies on a novel technique for identifying the largest congruence via successive approximations. This *proof technique* is of independent interest, as it has since been successfully re-used in several other process-algebraic settings involving preemption [11,12, 24].

To complete our theory of prioritized observation congruence we also provide an *axiomatization* of this behavioral relation for the class of finite processes and, along the way, extend the axiomatization of strong bisimulation in [10] to include the prioritization and deprioritization operators. Finally, we illustrate the utility of our behavioral congruence for system verification by means of an illustrative example, and discuss the influence of the prioritization and deprioritization operators in our setting on our results.

This article extends both an earlier conference paper [27] and a handbook chapter [13] by including deeper discussion on the context and implications of the work. Detailed proofs of all the results are also given, and the novel proof technique enabling the largest-congruence proof to be conducted is also highlighted.

1.3. Organization

The next section revisits the process algebra CCS^{prio} with priority introduced by Cleaveland and Hennessy in [10], as well as their behavioral congruence of strong bisimulation, and presents an axiomatization for finite processes with respect to all CCS^{prio} operators. Section 3 introduces a behavioral equivalence analogous to the observation equivalence of CCS, and Section 4 characterizes the largest congruence contained in it. An equational characterization of the observation congruence over finite processes is presented in Section 5, while Section 6 gives an example that applies this congruence to system verification. The last two sections discuss our approach in the light of related work and contain our conclusions and directions for future work, respectively.

2. CCS$^{\text{prio}}$ and prioritized strong bisimulation

This section reviews the language and equivalence given in [10] for modeling systems in which actions may be equipped with different priorities. We refer to the language as *Calculus of Communicating Systems with Priorities* (CCS$^{\text{prio}}$) and to the equivalence as *prioritized strong bisimulation*, as our setting is a conservative extension of Milner's CCS [25] that attaches priority values to actions. The reference article [10] also established a complete axiomatization of prioritized strong bisimulation for the sub-language of CCS$^{\text{prio}}$ that consists of the usual CCS operators only and leaves out the *prioritization* and *deprioritization* operators of CCS$^{\text{prio}}$. As the main technical contribution of this section we extend this axiomatization to the full CCS$^{\text{prio}}$ language.

2.1. CCS^{prio}: CCS with priorities

The language CCS$^{\text{prio}}$ provides constructs for building processes from atomic *actions*. Actions exhibit a two-level *priority structure*, with certain ones being designated as "prioritized" and others as "unprioritized". We will see later in Section 7 that this priority structure can easily be extended to a multi-level priority structure. However, even in the simpler setting of only two priority levels, all semantic and technical issues regarding the introduction of priority to process algebra can be illustrated. As in CCS, actions represent *potential* synchronizations that a process may be willing to engage in with its environment. Given a choice between a prioritized and an unprioritized synchronization, a process must choose the former. Hence, the ability of engaging in a prioritized synchronization *preempts* unprioritized behavior. We refer to this semantic concept as *global preemption* [13].

Formally, let Λ denote a countably infinite set of labels; intuitively, Λ contains the (unprioritized) "ports" that processes may synchronize over. Then the set of *unprioritized actions* A may be defined by $A =_{\text{df}} \Lambda \cup \overline{\Lambda} \cup \{\tau\}$, where $\overline{\Lambda} =_{\text{df}} \{\, \overline{\lambda} \mid \lambda \in \Lambda \,\}$. Action $\lambda \in \Lambda$ may be thought of as representing the receipt of an input on port λ, while $\overline{\lambda} \in \overline{\Lambda}$ constitutes the deposit of an output on λ; $\tau \notin \Lambda$ represents an internal (unprioritized) computation. To define the prioritized actions, let $\underline{\Lambda} =_{\text{df}} \{\, \underline{\lambda} \mid \lambda \in \Lambda \,\}$ be the set of "prioritized" ports, with $\overline{\underline{\Lambda}} =_{\text{df}} \{\, \overline{\underline{\lambda}} \mid \underline{\lambda} \in \underline{\Lambda} \,\}$. Then $\underline{A} =_{\text{df}} \underline{\Lambda} \cup \overline{\underline{\Lambda}} \cup \{\underline{\tau}\}$ is the set of *prioritized actions*, with $\underline{\tau}$ the *internal prioritized action*. We use $\mathcal{A} =_{\text{df}} A \cup \underline{A}$ to denote the set of all actions. In what follows we let $\alpha, \beta, \ldots$ range over $\mathcal{A}$, $a, b, c, \ldots$ over A, and $\underline{a}, \underline{b}, \underline{c}, \ldots$ over $\underline{A}$. We also use λ to represent elements in $\mathcal{A} \setminus \{\tau, \underline{\tau}\}$, μ as a representative of $A \setminus \{\tau\}$, and $\underline{\mu}$ as a representative of $\underline{A} \setminus \{\underline{\tau}\}$. We sometimes let the symbol '1' stand for either τ or $\underline{\tau}$. We also extend $\bar{}$ to all non-$\{\tau, \underline{\tau}\}$ actions by defining $\overline{\overline{\lambda}} =_{\text{df}} \lambda$, and if $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ then $\overline{L} =_{\text{df}} \{\, \overline{\lambda} \mid \lambda \in L \,\}$.

Terms in CCS$^{\text{prio}}$ are now defined by the following BNF, which coincides with that of CCS except for the addition of two new operators, the *prioritization operator* $\lceil\mu$ and the *deprioritization operator* $\lfloor\underline{\mu}$, as well as the use of a recursion operator instead of defining recursion via process identifiers.

$$t \;::=\; nil \mid \alpha.t \mid t + t \mid t|t \mid t\backslash\lambda \mid t[f] \mid t\lceil\mu \mid t\lfloor\underline{\mu} \mid x \mid fix(x : t)$$

Here, f is a *relabeling*, i.e., a mapping on $\mathcal{A}$ that preserves $\tau, \underline{\tau}$ and $\bar{}$ and satisfies $f(a) \in A$ and $f(\underline{a}) \in \underline{A}$. We also assume that relabelings f are such that the set $\{\, \lambda \in \mathcal{A} \setminus \{\tau, \underline{\tau}\} \mid \lambda \neq f(\lambda) \,\}$ is finite. Note, however, that we do not require that $f(\underline{a}) = \underline{f(a)}$. Moreover, x ranges over process variables, with recursion $fix(x : t)$ being the variable-binding operator. We use $\mathcal{E}$ to stand for the set of all terms of the language, and we adopt the usual definitions for *sort* of a term, *free* and *bound* variables, *open* and *closed* terms, *guarded recursion*, and *contexts*. In particular, the sort of a term contains all actions in which the term can engage according to the semantics defined below, and guards are visible actions but not τ and $\underline{\tau}$. We call the closed, guarded terms *processes*; $\mathcal{P}$ represents the set of all processes which is ranged over by $p, q, r, \ldots$. A term is *finite* if it contains no sub-term of the form $fix(x : t)$. For conciseness, we sometimes omit *nil*s from process terms and write, e.g., a for $a.nil$. Finally, we denote syntactic equality over terms by $\equiv$.

The operational semantics of processes is given as a transition relation $\rightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$; we write $p \xrightarrow{\alpha} q$ in lieu of $\langle p, \alpha, q\rangle \in \rightarrow$, as well as $p \xrightarrow{\alpha}$ if $p \xrightarrow{\alpha} q$ holds for some q, and $p \not\xrightarrow{\alpha}$ if not $p \xrightarrow{\alpha}$. The formal definition of $\rightarrow$ is given inductively in Table 1 and is adopted from [10]; note that even though the rules employ negative premises, the semantics is well-defined [10,32]. Intuitively, $p \xrightarrow{\alpha} q$ holds if p may engage in action α and thereafter behaves

Table 1
Operational semantics [10]

always		$\alpha.t \xrightarrow{\alpha} t$
$s \xrightarrow{a} s'$	$\Rightarrow$	$s + t \xrightarrow{a} s', \quad t + s \xrightarrow{a} s', \quad s\|t \xrightarrow{a} s'\|t, \quad t\|s \xrightarrow{a} t\|s'$
$s \xrightarrow{a} s',\ t \xrightarrow{\tau}\!\!\!\!\!\!\!\!/\ \ $	$\Rightarrow$	$s + t \xrightarrow{a} s', \quad t + s \xrightarrow{a} s'$
$s \xrightarrow{a} s',\ t \xrightarrow{\overline{a}} t'$	$\Rightarrow$	$s\|t \xrightarrow{\tau} s'\|t'$
$s \xrightarrow{a} s',\ s|t \xrightarrow{\tau}\!\!\!\!\!\!\!\!/\ \ $	$\Rightarrow$	$s\|t \xrightarrow{a} s'\|t, \quad t\|s \xrightarrow{a} t\|s'$
$s \xrightarrow{a} s',\ t \xrightarrow{\overline{a}} t',\ s|t \xrightarrow{\tau}\!\!\!\!\!\!\!\!/\ \ $	$\Rightarrow$	$s\|t \xrightarrow{\tau} s'\|t'$
$t \xrightarrow{\alpha} t',\ \alpha \neq \lambda, \overline{\lambda}$	$\Rightarrow$	$t\backslash\lambda \xrightarrow{\alpha} t'\backslash\lambda$
$s \xrightarrow{\mu} s'$	$\Rightarrow$	$s\lceil\mu \xrightarrow{\mu} s'\lceil\mu$
$s \xrightarrow{\alpha} s',\ \mu \neq \alpha$	$\Rightarrow$	$s\lceil\mu \xrightarrow{\alpha} s'\lceil\mu$
$s \xrightarrow{\underline{\mu}} s',\ s \xrightarrow{\tau}\!\!\!\!\!\!\!\!/\ \ $	$\Rightarrow$	$s\lfloor\underline{\mu} \xrightarrow{\underline{\mu}} s'\lfloor\underline{\mu}$
$s \xrightarrow{\underline{\mu}} s',\ s \xrightarrow{\tau}$	$\Rightarrow$	$s\lfloor\underline{\mu} \xrightarrow{\underline{\mu}} s'\lfloor\underline{\mu}$
$s \xrightarrow{\alpha} s',\ \underline{\mu} \neq \alpha$	$\Rightarrow$	$s\lfloor\underline{\mu} \xrightarrow{\alpha} s'\lfloor\underline{\mu}$
$t \xrightarrow{\alpha} t'$	$\Rightarrow$	$t[f] \xrightarrow{f(\alpha)} t'[f]$
$t[fix(x:t)/x] \xrightarrow{\alpha} t'$	$\Rightarrow$	$fix(x:t) \xrightarrow{\alpha} t'$

like q; the relative priorities of actions are represented by the fact that a process can engage in an unprioritized action only if it is *patient*, i.e., initially incapable of executing the prioritized internal action $\underline{\tau}$. One may wonder why, among the prioritized actions, only $\underline{\tau}$ has this preemptive power. The reason is that when $p \xrightarrow{\underline{\mu}} q$ holds, p is signaling its *potential* for synchronizing on $\underline{\mu}$, whereas $p \xrightarrow{\underline{\tau}} q$ denotes that a prioritized synchronization is indeed enabled for p.

Accordingly, process $\alpha.p$ may engage in α and then behave like p. The operator $+$ constitutes *non-deterministic choice*: $p + q$ may perform the prioritized actions of p or q and then behave like the process from which the action was chosen. Note that unprioritized actions are possible only if both p and q are patient. Process $p|q$ represents the *parallel composition* of p and q, with concurrent execution modeled via interleaving and synchronization on complementary actions resulting in the internal action τ, if the actions are unprioritized, and $\underline{\tau}$, otherwise. Note that $p|q$ may perform unprioritized actions only if $p|q$ is patient, which means that both p and q are patient *and* that there is no pending synchronization on a prioritized port shared by p and q. Operator $\backslash\lambda$ denotes *restriction* and delimits the scope for port λ, while $p[f]$ is a *relabeling* where actions of p are relabeled according to the mapping f on actions. Process $p\lceil\mu$ *prioritizes* the (unprioritized) μ actions in which p may engage; note that p may only engage in such actions if it is patient. Similarly, $p\lfloor\underline{\mu}$ *deprioritizes* $\underline{\mu}$ actions, but only if in so doing, the resulting μ actions remain possible, i.e., only if p is patient. Finally, $fix(x:t)$ represents a recursively defined process that is a distinguished solution to the equation $x = t$.

The operational semantics for CCS^{prio} possesses several desired properties [10]. First, the operational rules involving only prioritized actions coincide with those of CCS. Second, our semantics correctly encodes our intuition of global preemption since $p \xrightarrow{\underline{\tau}}$ *implies* $p \xrightarrow{a}\!\!\!\!\!\!\!\!/\ \ $, for any process p and unprioritized action a. Third, the sort of any process is finite; this is a consequence of only allowing *finite* summation as well as *finite* relabelings that satisfy $|\{\ \lambda \in \mathcal{A} \setminus \{\tau, \underline{\tau}\} \mid \lambda \neq f(\lambda)\ \}| < \infty$.

As in [10], we adopt Milner's notion of strong bisimulation [25] and refer to it as *prioritized strong bisimulation*.

Definition 1 (*Prioritized strong bisimulation [10]*).
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *strong prioritized bisimulation relation* if, for all $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$:

(1) $p \xrightarrow{\alpha} p'$ implies $\exists q'.\, q \xrightarrow{\alpha} q'$ and $\langle p', q'\rangle \in \mathcal{R}$.
(2) $q \xrightarrow{\alpha} q'$ implies $\exists p'.\, p \xrightarrow{\alpha} p'$ and $\langle p', q'\rangle \in \mathcal{R}$.

We write $p \simeq q$ if $\langle p, q\rangle \in \mathcal{R}$ for some prioritized strong bisimulation relation $\mathcal{R}$, and call $\simeq$ *prioritized strong bisimulation*.

Table 2
Axioms for finite processes without $\lceil\mu$ and $\lfloor\underline{\mu}$ operators [10]

A1	$x + x = x$
A2	$x + y = y + x$
A3	$x + (y + z) = (x + y) + z$
A4	$x + nil = x$
P	$a.x + \underline{\tau}.y = \underline{\tau}.y$
INT	Let p and q denote $\sum \alpha_i.p_i$ and $\sum \beta_j.q_j$, respectively. Then $p\|q = \sum \alpha_i.(p_i\|q) + \sum \beta_j.(p\|q_j) + \sum_{\alpha_i=\overline{\beta_j}\in A} \tau.(p_i\|q_j) + \sum_{\alpha_i=\overline{\beta_j}\in \underline{A}} \underline{\tau}.(p_i\|q_j)$
RES1	$nil\backslash\lambda = nil$
RES2	$(\alpha.x)\backslash\lambda = \begin{cases} nil & \text{if } \alpha \in \{\lambda, \overline{\lambda}\} \\ \alpha.(x\backslash\lambda) & \text{otherwise} \end{cases}$
RES3	$(x + y)\backslash\lambda = (x\backslash\lambda) + (y\backslash\lambda)$
REL1	$nil[f] = nil$
REL2	$(\alpha.x)[f] = f(\alpha).(x[f])$
REL3	$(x + y)[f] = x[f] + y[f]$

Table 3
Axioms for $\lceil\mu$ and $\lfloor\underline{\mu}$

$\lceil 1$	$nil\lceil\mu$	=	nil	
$\lceil 2$	$(\alpha.x)\lceil\mu$	=	$\begin{cases} \underline{\mu}.(x\lceil\mu) & \text{if } \alpha = \mu \\ \underline{\alpha}.(x\lceil\mu) & \text{otherwise} \end{cases}$	
$\lceil 3$	$(x + \underline{\tau}.y + \underline{a}.z)\lceil\mu$	=	$(x + \underline{\tau}.y)\lceil\mu + \underline{a}.(z\lceil\mu)$	
$\lceil 4$	$(x + \nu_1.y + \nu_2.z)\lceil\mu$	=	$(x + \nu_1.y)\lceil\mu + (x + \nu_2.z)\lceil\mu$	$\nu_1, \nu_2 \in \mathcal{A} \setminus \{\underline{\tau}\}$
$\lfloor 1$	$nil\lfloor\underline{\mu}$	=	nil	
$\lfloor 2$	$(\alpha.x)\lfloor\underline{\mu}$	=	$\begin{cases} \mu.(x\lfloor\underline{\mu}) & \text{if } \alpha = \underline{\mu} \\ \alpha.(x\lfloor\underline{\mu}) & \text{otherwise} \end{cases}$	
$\lfloor 3$	$(x + \underline{\tau}.y + \underline{a}.z)\lfloor\underline{\mu}$	=	$(x + \underline{\tau}.y)\lfloor\underline{\mu} + \underline{a}.(z\lfloor\underline{\mu})$	
$\lfloor 4$	$(x + \nu_1.y + \nu_2.z)\lfloor\underline{\mu}$	=	$(x + \nu_1.y)\lfloor\underline{\mu} + (x + \nu_2.z)\lfloor\underline{\mu}$	$\nu_1, \nu_2 \in \mathcal{A} \setminus \{\underline{\tau}\}$

It is easy to see that $\simeq$ is itself a prioritized strong bisimulation relation and that it is indeed the largest such relation. The fact that $\simeq$ is a congruence for CCS$^{\text{prio}}$ was already proved in [10]; it can also be immediately inferred when inspecting the format of our operational rules in Table 1 [32].

2.2. Axiomatizing prioritized strong bisimulation

We now give an equational axiomatization of $\simeq$ for the class of *finite* processes. Cleaveland and Hennessy proved in [10] that the equations of Table 2 provide a sound and complete axiomatization for the CCS$^{\text{prio}}$ sublanguage that does not include the prioritization operator $\lceil\mu$ and the deprioritization operator $\lfloor\underline{\mu}$. The axioms of Table 2 are exactly the ones for strong bisimulation in CCS[25], with the addition of Axiom P which encodes our semantic concept of global preemption.

In order to cover all finite processes we extend the axiomatization of Table 2 with the axioms in Table 3, the first four of which axiomatize the prioritization operator $\lceil\mu$ and the last four of which axiomatize the deprioritization operator $\lfloor\underline{\mu}$. While Axioms $\lceil 1$ and $\lfloor 1$ are trivial, Axioms $\lceil 2$ and $\lfloor 2$ reflect our basic intuition of prioritizing and deprioritizing an action, respectively. Note that Axioms $\lceil 3$ and $\lceil 4$ cannot be replaced by a single axiom of the form "$(x + y)\lceil\mu = x\lceil\mu + y\lceil\mu$" as, e.g., $(\underline{\tau} + \mu)\lceil\mu \not\simeq \underline{\tau} + \underline{\mu}$. This is because action μ may only be prioritized within a process if the process is patient, according to our operational rules of Table 1. Similarly, Axioms $\lfloor 3$ and $\lfloor 4$ cannot be replaced by an axiom of the form "$(x + y)\lfloor\underline{\mu} = x\lfloor\underline{\mu} + y\lfloor\underline{\mu}$" as, e.g., $(\underline{\tau} + \mu)\lfloor\underline{\mu} \not\simeq \underline{\tau} + \mu$.

Let $E_{\simeq}$ denote the set of axioms provided in Tables 2 and 3. The rest of this section is devoted to the proof that $E_{\simeq}$ completely characterizes prioritized strong bisimulation over finite processes. The following theorem states the soundness of our axioms.

Theorem 2 (Soundness)**.**
Let p and q be finite processes. Then, $p \simeq q$ if $\vdash_{E_\simeq} p = q$.

Proof. The soundness of the axioms presented in Table 2 is proved in [10]. The soundness of the new axioms of Table 3 is proved in a straightforward manner by constructing prioritized strong bisimulation relations containing the appropriate pair of processes. More precisely, if p and q are closed terms obtained by instantiating the left-hand side and right-hand side of some axiom given in Table 3, then the relation $\{\langle p, q\rangle\} \cup \{\ \langle r, r\rangle \mid r \in \mathcal{P}\ \}$ is a prioritized strong bisimulation relation. The proof details are straightforward. □

Now we establish completeness, i.e., if p and q are finite processes such that $p \simeq q$, then $\vdash_{E_\simeq} p = q$. The proof of this statement involves the notion of *summation form* and *normal form* introduced in [10], which we recall first.

Definition 3 (*Summation form and normal form [10]*)**.**
The process $\sum_{i\in I} \alpha_i.p_i$, for a finite index set I, is defined to be a *summation form* if each p_i, for $i \in I$, is also a summation form. The empty sum, i.e., $I = \emptyset$, is identified with *nil*. If $p \equiv \sum \alpha_i.p_i$ is a summation form, then the *depth* $d(p)$ of term p is defined inductively as follows: $d(nil) = 0$ and $d(\sum_{i\in I} \alpha_i.p_i) = 1 + max\{\ d(p_i) \mid i \in I\ \}$.

A summation form $\sum_{i\in I} \alpha_i.p_i$ is defined to be a *normal form* if (i) $\alpha_i \equiv \underline{\tau}$ for some $i \in I$ implies that $\alpha_j \notin A$, for any $j \neq i$, and if (ii) each p_i, for $i \in I$, is also a normal form.

Intuitively, the depth $d(p)$ of a summation form p denotes the length of the longest sequence of actions it can execute. Since any summation form is a finite process, its depth is well defined. The notion of depth is needed when showing that every finite process p can be rewritten as a normal form, i.e., there exists a finite process p' such that $\vdash p = p'$. This will be done by induction on the number k of prioritization or deprioritization constructors present in a given finite process p. In the base case, k is 0 and p does not contain any $\lceil$- or $\lfloor$-constructors, and the procedure for rewriting p into a normal form using the axioms presented in Table 2 is illustrated in [10]. For the induction case, i.e., $k > 0$, p contains a subterm of the form $r\lceil\mu$ or $r\lfloor\underline{\mu}$, where r is free of $\lceil$- or $\lfloor$-operators. The following lemma illustrates how such a subterm can be rewritten as a normal form using only axioms of our proof system $E_\simeq$.

Lemma 4 (Rewriting $\lceil$- and $\lfloor$-processes into normal forms)**.**
If r is a finite process that does not contain any constructors of type $\lceil$ or $\lfloor$, then it is possible to rewrite $r\lceil\mu$ and $r\lfloor\underline{\mu}$ as normal forms using $E_\simeq$, for any $\mu \in A$ and $\underline{\mu} \in \underline{A}$.

Proof. Consider the term $r\lceil\mu$ for some finite process r not containing any prioritization and deprioritization constructors, and some unprioritized action μ. We know from [10] that there exists a normal form $r' \equiv \sum_{i=1}^{n} \alpha_i.r_i$ such that r can be rewritten as r' using the proof system $E_\simeq$. Let d be the depth of $\sum_{i=1}^{n} \alpha_i.r_i$. We show how the term $\left(\sum_{i=1}^{n} \alpha_i.r_i\right)\lceil\mu$ can be rewritten as a normal form using the proof system $E_\simeq$, by inducting upon d.

- *Base case:* $d = 0$.
  Hence $\left(\sum_{i=1}^{n} \alpha_i.r_i\right)\lceil\mu \equiv nil\lceil\mu$. By Axiom $\lceil$1, term $nil\lceil\mu$ can be rewritten as *nil*, which is a normal form.
- *Induction step:* $d > 0$.
  Here, we induct upon n. Since $d > 0$, it is necessarily the case that $n > 0$. For the base case for inducting upon n, we have $n = 1$. But $\left(\sum_{i=1}^{1} \alpha_i.r_i\right)\lceil\mu \equiv (\alpha_1.r_1)\lceil\mu$, which can be rewritten as $\beta_1.(r_1\lceil\mu)$ by Axiom $\lceil$2, where β_1 is $\underline{\alpha_1}$ or α_1 depending upon whether $\alpha_1 = \mu$ or not. Since r_1 is a normal form and since its depth is strictly less than that of $\alpha_1.r_1$, the induction hypothesis based on d is applicable to $r_1\lceil\mu$, i.e., r_1 can be rewritten as a normal form n_1 using $E_\simeq$. Thus $(\alpha_1.r_1)\lceil\mu$ can be rewritten as the normal form $\beta_1.n_1$.
  For the induction case for inducting upon n, we have $n \geq 2$. Then the term r' can be rewritten as $r'' + \alpha_{n-1}.r_{n-1} + \alpha_n.r_n$, where $r'' \equiv \sum_{i=1}^{n-2} \alpha_i.r_i$. If $n > 2$ then r' is already in this form; however, if $n = 2$, then take $r'' \equiv nil$. The proof now splits into two cases depending upon whether any of α_{n-1} and α_n is the action $\underline{\tau}$.

  (1) Suppose that at least one of α_{n-1} and α_n is $\underline{\tau}$; w.l.o.g. (cf. Axiom A2), assume $\alpha_{n-1} \equiv \underline{\tau}$.
  Since the summation form $\sum_{i=1}^{n} \alpha_i.r_i$ is also a normal form, α_n cannot be an unprioritized action. Thus $r'\lceil\mu$ is an instantiation of the left-hand side of Axiom $\lceil$3 and can be rewritten as $(r'' + \underline{\tau}.r_{n-1})\lceil\mu +$

$\alpha_n.(r_n \lceil \mu)$. Both terms $r'' + \underline{\tau}.r_{n-1}$ and r_n are normal forms, with the former term having strictly fewer summands than that of r', and the latter term having strictly less depth than r'.
Hence, the induction hypothesis based upon n is applicable to $(r' + \underline{\tau}.r_{n-1})\lceil\mu$ and that upon d is applicable to $r_n\lceil\mu$, i.e., $(r' + \underline{\tau}.r_{n-1})\lceil\mu$ and $r_n\lceil\mu$ can be rewritten as normal forms n_1 and n_2, respectively. Thus, $(\sum_{i=1}^{n} \alpha_i.r_i)\lceil\mu$ can be rewritten as the summation form $n_1 + \alpha_n.n_2$. It is not difficult to see that this summation form is also a normal form.

(2) Suppose that both α_{n-1} and α_n belong to $\mathcal{A} \setminus \{\underline{\tau}\}$.
In this case, a similar procedure as the one used in Case (1) above can be used to rewrite $(\sum_{i=1}^{n} \alpha_i.r_i)\lceil\mu$ into a normal form, this time using Axiom $\lceil 4$ instead of $\lceil 3$.

This concludes the proof of the induction cases for both n and d.

The procedure for handling the term $r\lfloor\underline{\mu}$ is similar and makes use of Axioms $\lfloor 1$ through $\lfloor 4$. □

Hence, every finite process can be rewritten into normal form. In order to prove our proof system $E_{\simeq}$ complete we can now simply refer to the following theorem of [10].

Theorem 5 (from [10])**.**
Let p and q be finite processes in normal form. Then $p \simeq q$ implies $\vdash_{E_{\simeq}} p = q$.

We thus obtain the desired completeness result as a corollary to Lemma 4 and Thm. 5.

Corollary 6 (Completeness)**.**
Let p and q be finite processes. Then $p \simeq q$ implies $\vdash_{E_{\simeq}} p = q$.

The establishment of a sound and complete axiomatization of prioritized strong bisimulation for the *full* CCS^{prio} language completes the semantic theory for $\simeq$. It is worth noting here that the above axiomatization can be generalized to arbitrary processes, i.e., processes potentially containing recursion, using the standard technique proposed by Milner in [26].

3. Prioritized observation equivalence and congruence

Prioritized strong bisimulation $\simeq$ treats the internal actions τ and $\underline{\tau}$ no differently from the other actions. Equivalent processes must therefore exhibit the same "levels" of internal behavior, which complicates the use of $\simeq$ for system verification. For example, processes $a.\underline{\tau}.b$ and $a.b$ are *not* related by $\simeq$. However, to an external observer, they should be indistinguishable since $\underline{\tau}$ is an internal action. Thus, $\simeq$ may be seen as too discriminating. The aim of this section is to define a congruence over processes which abstracts away from *internal* computation and thus relates processes if and only if they have the same *external* behavior.

Our approach follows that of Milner in [25] for the definition of observational congruence. We first define a straightforward modification of $\simeq$ by introducing a new transition relation $\Rightarrow_{\text{n}}$ that "absorbs" internal computation steps. This naïve equivalence turns out not to be a congruence, and we spent significant effort on repairing the compositionality flaws, thereby establishing a congruence called *prioritized observation congruence*. The main challenge lies in proving compositionality of prioritized observation congruence for parallel composition. In the next section we further prove that this congruence is a distinguished one: it is the largest congruence contained in the aforementioned naïve equivalence and is in this sense as insensitive to internal computation as possible.

To define this naïve reference relation we introduce the following definitions.

Definition 7 (*Naïve weak transition relation*)**.**
Let $\alpha \in \mathcal{A}$. Then,

(1) $\stackrel{\epsilon}{\Rightarrow}_{\text{n}} =_{\text{df}} (\stackrel{\tau}{\rightarrow} \cup \stackrel{\underline{\tau}}{\rightarrow})^*$.
(2) $\stackrel{\alpha}{\Rightarrow}_{\text{n}} =_{\text{df}} \stackrel{\epsilon}{\Rightarrow}_{\text{n}} \circ \stackrel{\alpha}{\rightarrow} \circ \stackrel{\epsilon}{\Rightarrow}_{\text{n}}$.
(3) $\hat{\alpha} =_{\text{df}} \epsilon$ if $\alpha \in \{\tau, \underline{\tau}\}$, and α otherwise.

Intuitively, $p \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} q$ if p may engage in a sequence of prioritized and unprioritized internal transitions and evolve to q, while $p \stackrel{\alpha}{\Rightarrow}_{\mathrm{n}} q$ holds if p may engage in internal computation, then α, and then more internal computation before evolving to q. The notation $\hat{\alpha}$ represents the "visible content" of action α. We now define *naïve prioritized weak bisimulation* as follows.

Definition 8 (*Naïve prioritized weak bisimulation*)**.**
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called a *naïve prioritized weak bisimulation relation* if, for every $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, the following conditions hold:

- $p \stackrel{\alpha}{\rightarrow} p'$ implies $\exists q'.\, q \stackrel{\hat{\alpha}}{\Rightarrow}_{\mathrm{n}} q'$ and $\langle p', q'\rangle \in \mathcal{R}$.
- $q \stackrel{\alpha}{\rightarrow} q'$ implies $\exists p'.\, p \stackrel{\hat{\alpha}}{\Rightarrow}_{\mathrm{n}} p'$ and $\langle p', q'\rangle \in \mathcal{R}$.

We write $p \approx_{\mathrm{n}} q$ if $\langle p, q\rangle \in \mathcal{R}$ for some naïve prioritized weak bisimulation relation $\mathcal{R}$, and call $\approx_{\mathrm{n}}$ *naïve prioritized weak bisimulation.*

It is straightforward to establish that $\approx_{\mathrm{n}}$ is the largest naïve prioritized weak bisimulation relation and that it is an equivalence on processes. In order for an equivalence to be a satisfactory semantic relation, it is imperative that the relation is a congruence with respect to all the process constructors present in the language. This is due to the fact that compositional reasoning is the main tool for system verification in any process-algebraic framework. Unfortunately, $\approx_{\mathrm{n}}$ is *not* a congruence. Unsurprisingly, and analogously to weak bisimulation in CCS[25], it is not preserved by the non-deterministic choice operator $+$. More disturbingly, $\approx_{\mathrm{n}}$ is also not preserved by parallel composition. To see this, consider the following examples; in each case $p \approx_{\mathrm{n}} q$ but $p|r \not\approx_{\mathrm{n}} q|r$.

Example 9 (*Non-preservation of Milner's tau-laws*)**.**

(1) $p \equiv \underline{\mu}.\tau.\underline{\mu}$, $q \equiv \underline{\mu}.\underline{\mu}$, $r \equiv fix(x : \underline{\tau}.x)$.
(2) $p \equiv \underline{a} + \tau.\underline{a}$, $q \equiv \tau.\underline{a}$, $r \equiv \overline{a} + \mu$.
(3) $p \equiv a.(\underline{b} + \tau.\mu)$, $q \equiv a.(\underline{b} + \tau.\mu) + a.\mu$, $r \equiv \overline{b} + \underline{\mu}$.
(4) $p \equiv nil$, $q \equiv fix(x : \underline{\tau}.x)$, $r \equiv \mu$.

In the first example, $q|r \stackrel{\mu}{\rightarrow} \underline{\mu}|r \stackrel{\mu}{\rightarrow}$, while no $\stackrel{\mu}{\Rightarrow}_{\mathrm{n}}$-derivative of $p|r$ can perform action $\underline{\mu}$; note that this shows that the first tau-law of Milner [25] is in general invalid in our setting. In the second example, $q|r \stackrel{\mu}{\rightarrow}$ whereas $p|r \stackrel{\mu}{\Rightarrow}_{\mathrm{n}}$; this shows that Milner's second tau-law is in general unsound here. In the third example, $q|r$ can perform action a to reach a process in which both actions μ and $\underline{\mu}$ are enabled. However, $p|r$ can never evolve to an equivalent process, whence Milner's third tau-law does not hold in our setting. Finally, in the fourth example, $p|r$ can engage in action μ, but $q|r$ can never reach a process that enables μ.

One reason that parallel composition does not preserve $\approx_{\mathrm{n}}$ is that $p \approx_{\mathrm{n}} q$ may hold even though p may reach a patient process through $\underline{\tau}$-transitions while q cannot (see Ex. 9(4)). Another reason (see Exs. 9(1–3)) is that, although q and r may both be patient, their parallel composition $q|r$ may not be because of the possibility of synchronizations between them involving prioritized actions. Thus, it may be the case that when $p \approx_{\mathrm{n}} q$, an α-transition of p can only be matched by a naïve weak $\hat{\alpha}$-transition that includes some τ-transitions. When put in parallel with r, however, potential prioritized synchronizations between some derivative of q and r might prevent the possibility of these τ-transitions of q. Therefore, in order to arrive at a suitable congruence contained in $\approx_{\mathrm{n}}$, we must take account of the possibility of reaching patient processes through $\underline{\tau}$-transitions and of the preemptability of τ-transitions. With this motivation, we introduce the following definitions.

Definition 10. Let p be a process.

(1) $I(p) =_{\mathrm{df}} \{\, \lambda \in \mathcal{A} \setminus \{\tau, \underline{\tau}\} \mid p \stackrel{\lambda}{\rightarrow} \,\}$
(2) $p\!\downarrow\!\downarrow$ if there exists a patient process q such that $p \stackrel{\tau}{\rightarrow}^{*} q$.

Intuitively, $I(p)$ is the set of visible actions that p may initially perform. We also use $I_A(p) =_{\mathrm{df}} I(p) \cap A$ and $I_{\underline{A}}(p) =_{\mathrm{df}} I(p) \cap \underline{A}$ to denote the visible unprioritized and prioritized actions initially available to p. Moreover, $p\!\downarrow\!\downarrow$ means that it is *possible* for process p to evolve to a patient process through a sequence of zero or more $\underline{\tau}$-transitions. Note that this definition differs from the notion of *convergence* that one often finds in the literature [15]: here, if $p\!\downarrow\!\downarrow$ then it is still possible for p to engage in an infinite sequence of $\underline{\tau}$-transitions.

Definition 11 (*Prioritized weak transition relation*)**.**

(1) $\stackrel{\epsilon}{\Rightarrow} =_{\mathrm{df}} \stackrel{\underline{\tau}}{\rightarrow}^*$; we re-define $\hat{\underline{\tau}} =_{\mathrm{df}} \underline{\epsilon}$.
(2) Let $\alpha \in \mathcal{A} \setminus \{\tau\}$. Then $\stackrel{\alpha}{\Rightarrow} =_{\mathrm{df}} \stackrel{\epsilon}{\Rightarrow} \circ \stackrel{\alpha}{\rightarrow} \circ \stackrel{\epsilon}{\Rightarrow}$.
(3) Let $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$. Then $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$ is defined inductively: $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$ if
- $p \equiv q$, or
- $\exists r.\ p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} r \stackrel{\underline{\tau}}{\rightarrow} q$, or
- $\exists r.\ p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} r \stackrel{\underline{\tau}}{\rightarrow} q$ and $I(r) \subseteq L$.

We write $p \stackrel{\tau}{\underset{L}{\Longrightarrow}} q$ if $\exists p', q'.\ p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} p' \stackrel{\underline{\tau}}{\rightarrow} q' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$ and $I(p') \subseteq L$.

Intuitively, $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$ holds if there exists a sequence $p_0, \ldots, p_n$ of processes such that $p \equiv p_0 \stackrel{1}{\rightarrow} p_1 \stackrel{1}{\rightarrow} \cdots \stackrel{1}{\rightarrow} p_n \equiv q$ and, if p_i is patient, then the initial visible actions available to p_i must be contained in L, for any $0 \leq i \leq n$. For example, $\underline{\tau}.(\underline{b} + \tau.c) \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} c$ holds for those L containing action $\underline{b}$.

The next lemma remarks on several straightforward properties of $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$ that follow immediately from Def. 11.

Lemma 12 (Properties of prioritized weak transitions)**.**
For any $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ and $p, q \in \mathcal{P}$ the following are true:

(1) $\stackrel{\tau}{\underset{L}{\Longrightarrow}} \circ \stackrel{\epsilon}{\Rightarrow} = \stackrel{\epsilon}{\Rightarrow} \circ \stackrel{\tau}{\underset{L}{\Longrightarrow}} = \stackrel{\tau}{\underset{L}{\Longrightarrow}}$.
(2) *If* $p \stackrel{\epsilon}{\Rightarrow} q$ *then* $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$.
(3) *If* $p \stackrel{\underline{\tau}}{\rightarrow} q$ *and* $I(p) \subseteq L$, *then* $p \stackrel{\tau}{\underset{L}{\Longrightarrow}} q$.
(4) $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}} \subseteq \stackrel{\epsilon}{\underset{M}{\Longrightarrow}}$ *if* $L \subseteq M$.

To understand the importance of relation $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$, suppose that $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$ and further that some process r is patient and incapable of synchronizing with any prioritized action in L. We can then assert that $p|r \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} q|r$. This fact is a consequence of the following lemma.

Lemma 13. *Assume $p, q, r \in \mathcal{P}$ and $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ are such that (i) $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$, (ii) r is patient, and (iii) $I_{\underline{A}}(r) \cap \overline{L} = \emptyset$. Then, $p|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} q|r$ for $M =_{df} L \cup I(r)$.*

Proof. The proof is by induction on the number of 1-transitions involved in $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$. For the base case we have $p \equiv q$, and by using the first point of Def. 11(3) we can infer that $p|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} q|r$. For the induction case, there exists some process p_1 such that one of the following two conditions holds for some p_1: (i) $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} p_1 \stackrel{\underline{\tau}}{\rightarrow} q$, or (ii) $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} p_1 \stackrel{\underline{\tau}}{\rightarrow} q$ and $I(p_1) \subseteq L$.

Suppose the first condition holds. Then, by induction hypothesis, we have $p|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} p_1|r$, and from the operational semantics of the parallel composition operator we have $p_1|r \stackrel{\tau}{\rightarrow} q|r$. The required result that $p|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} q|r$ now follows from Lemma 12(1).

Now suppose the second condition holds. Again by induction hypothesis we have $p|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} p_1|r$. But in order to infer $p_1|r \stackrel{\tau}{\rightarrow} q|r$ we have additionally to show that $p_1|r$ is patient. This is true because it is given that r is patient and that $I_{\underline{A}}(r) \cap \overline{I_{\underline{A}}(p_1)}$ is empty. The latter statement follows from the facts that $I(p_1) \subseteq L$ and $I_{\underline{A}}(r) \cap \overline{L} = \emptyset$. Thus, we have $p_1|r \stackrel{\tau}{\rightarrow} q|r$, so that $p_1|r \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} q|r$ by Lemma 12(3). The required result then follows from the transitivity of $\stackrel{\epsilon}{\underset{M}{\Longrightarrow}}$. □

One may also wonder why the definition of $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q$ constrains the unprioritized as well as the prioritized actions of the patient intermediate states between p and q, i.e., why "$I(r) \subseteq L$" rather than "$I_{\underline{A}}(r) \subseteq L$" is used in the definition. This requirement is needed to ensure that the equivalence we are about to define is a congruence for the prioritization operator $\lceil\mu$. We will show in Sec. 7.1 that it can be dropped when the prioritization constructor is left out.

We may now introduce our notion of *prioritized observation equivalence*; it will repair the compositionality bug of the naïve prioritized weak bisimulation with respect to parallel composition.

Definition 14 (*Prioritized observation equivalence*)**.**
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called a *prioritized weak bisimulation relation*, or pwb for short, if, for all $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, the following conditions hold:

(1) $p\!\downarrow\!\downarrow$ if and only if $q\!\downarrow\!\downarrow$.

(2) $p \stackrel{\alpha}{\rightarrow} p'$ implies $\begin{cases} \exists q'.\, q \stackrel{\hat{\alpha}}{\Rightarrow} q' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha \neq \tau \\ \exists q'.\, q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha = \tau \end{cases}$

(3) $q \stackrel{\alpha}{\rightarrow} q'$ implies $\begin{cases} \exists p'.\, p \stackrel{\hat{\alpha}}{\Rightarrow} p' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha \neq \tau \\ \exists p'.\, p \stackrel{\epsilon}{\underset{I(q)}{\Longrightarrow}} p' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha = \tau \end{cases}$

We write $p \approxeq q$ if $\langle p, q\rangle \in \mathcal{R}$ for some prioritized weak bisimulation relation $\mathcal{R}$, and call $\approxeq$ *prioritized observation equivalence*.

It is straightforward to establish that $\approxeq$ is the largest pwb and that it is also an equivalence. In addition, we have that $\simeq$ is strictly contained in $\approxeq$. While part of the aforementioned compositionality bug regarding parallel composition is addressed by using the parameterized transition relation when matching τ-transitions (cf. Cond. (2)), the other part is dealt with by including Cond. (1). The next lemma states several important properties enjoyed by pwb's, which will be used in the proofs of our later results.

Lemma 15. *Let $\mathcal{R}$ be a pwb with $\langle p, q\rangle \in \mathcal{R}$, and let $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$. Then*

(1) $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} p'$ *implies* $\exists q'.\, q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} q'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.
(2) $p \stackrel{\epsilon}{\Rightarrow} p'$ *implies* $\exists q'.\, q \stackrel{\epsilon}{\Rightarrow} q'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.
(3) *p is patient implies* $\exists q'.\, q \stackrel{\epsilon}{\Rightarrow} q'$, $\langle p, q'\rangle \in \mathcal{R}$, *and q' is patient*.
(4) *p is patient implies* $I(q) \subseteq I(p)$.
(5) $p \stackrel{\epsilon}{\Rightarrow} p'$ *and p' is patient implies* $\exists q'.\, q \stackrel{\epsilon}{\Rightarrow} q'$, $\langle p', q'\rangle \in \mathcal{R}$, *and q' is patient*.

Proof. The proofs of Parts (1) and (2) follow by induction on the number of l-transitions involved in $p \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} p'$ and $p \stackrel{\epsilon}{\Rightarrow} p'$, respectively, and use Lemma 12.

To prove Part (3), observe that the patience of p implies that $p\!\downarrow\!\downarrow$. Then, by the first condition of Def. 14, we have $q\!\downarrow\!\downarrow$. Thus, by definition of predicate $\downarrow\!\downarrow$, we infer the existence of some process q' that is patient and satisfies $q \stackrel{\epsilon}{\Rightarrow} q'$. Using Part (2) we can deduce the existence of a process p' such that $p \stackrel{\epsilon}{\Rightarrow} p'$ and $\langle p', q'\rangle \in \mathcal{R}$. Since p is patient, it is necessarily the case that $p' \equiv p$, which is the required result.

To establish Part (4), let $\lambda \in I(q)$. This means that $q \stackrel{\lambda}{\rightarrow} q'$ for some process q'. By using the third condition of Def. 14 we can infer the existence of a process p' such that $p \stackrel{\lambda}{\Rightarrow} \circ \stackrel{\tau}{\rightarrow}{}^{*} p'$ and $\langle p', q'\rangle \in \mathcal{R}$. But p is patient. This means that $p \stackrel{\lambda}{\rightarrow}$ which implies $\lambda \in I(p)$. Thus, we have $I(q) \subseteq I(p)$.

Now we prove Part (5). Using Part (2) we infer the existence of some process q'' such that $q \stackrel{\epsilon}{\Rightarrow} q''$ and $\langle p', q''\rangle \in \mathcal{R}$. But p' is patient and therefore $p'\!\downarrow\!\downarrow$. Since $\mathcal{R}$ is a pwb, we have $q''\!\downarrow\!\downarrow$. Consequently, there exists a patient process q' such that $q'' \stackrel{\epsilon}{\Rightarrow} q'$. Again by using Part (2), we know of some process p'' such that $p' \stackrel{\epsilon}{\Rightarrow} p''$ and $\langle p'', q'\rangle \in \mathcal{R}$. Since p' is patient, it is necessarily the case that $p' \equiv p''$. □

The following lemma states that $\approxeq$ is preserved under all CCS$^{\text{prio}}$ operators but non-deterministic choice and recursion.

Lemma 16 (Compositionality)**.**
Let $p, q, r \in \mathcal{P}$ such that $p \approxeq q$, let $\lambda \in \Lambda, \alpha \in \mathcal{A}, \mu \in A \setminus \{\tau\}$, and let f be a relabeling. Then the following properties hold:

(1)	$\alpha.p \approxeq \alpha.q$	(3)	$p\backslash\lambda \approxeq q\backslash\lambda$	(5)	$p\lceil\mu \approxeq q\lceil\mu$
(2)	$p\|r \approxeq q\|r$	(4)	$p[f] \approxeq q[f]$	(6)	$p\lfloor\underline{\mu} \approxeq q\lfloor\underline{\mu}$

Proof. We only prove $p|r \approxeq q|r$ whenever $p \approxeq q$. The proofs of the other parts of the lemma are either similar to the corresponding ones presented in [25] or, in the case of (5) and (6), are easy. Let $\mathcal{R} =_{\text{df}} \{\, \langle p|r, q|r\rangle \mid p \approxeq q,\ r \in \mathcal{P} \,\}$. Since $\approxeq$ is an equivalence, it is symmetric, and this implies that $\mathcal{R}$ is also symmetric. Thus, to prove that $\mathcal{R}$ is a pwb it suffices to show that, when $\langle \overline{p}, \overline{q}\rangle \in \mathcal{R}$, the following is true:

(1) $\overline{p} \stackrel{\alpha}{\rightarrow} \overline{p}'$ implies that, for $\alpha \neq \tau, \exists \overline{q}'.\, \overline{q} \stackrel{\hat{\alpha}}{\Rightarrow} \overline{q}'$ and $\langle \overline{p}', \overline{q}'\rangle \in \mathcal{R}$, and that, for $\alpha = \tau, \exists \overline{q}'.\, \overline{q} \stackrel{\epsilon}{\Longrightarrow}_{I(\overline{p})} \overline{q}'$ and $\langle \overline{p}', \overline{q}'\rangle \in \mathcal{R}$.
(2) $\overline{p}\!\downarrow\!\downarrow$ implies $\overline{q}\!\downarrow\!\downarrow$.

To establish Cond. (1), suppose $\langle \overline{p}, \overline{q}\rangle \in \mathcal{R}$. Then, there exist processes p, q and r such that $\overline{p} \equiv p|r$, $\overline{q} \equiv q|r$ and $p \approxeq q$. If $\overline{p} \stackrel{\alpha}{\rightarrow} \overline{p}'$ and $\alpha \in \mathcal{A} \setminus \{\tau\}$, then, by the usual arguments, we can show the existence of some $\overline{q}'$ such that $\overline{q} \stackrel{\hat{\alpha}}{\Rightarrow} \overline{q}'$ and $\langle \overline{p}', \overline{q}'\rangle \in \mathcal{R}$.

Now suppose $\overline{p} \stackrel{\tau}{\rightarrow} \overline{p}'$, whence $p|r$ is patient. This implies that p and r are patient and that $I_{\underline{A}}(r) \cap \overline{I_{\underline{A}}(p)} = \emptyset$. We must find a process $\overline{q}$ such that $\overline{q} \stackrel{\epsilon}{\Longrightarrow}_{I(\overline{p})} \overline{q}'$ and $\langle \overline{p}', \overline{q}'\rangle \in \mathcal{R}$. The proof splits into three cases:

(1) $p \stackrel{\tau}{\rightarrow} p'$ and $\overline{p}' \equiv p'|r$.
Since $p \approxeq q$, there must be some q' such that $q \stackrel{\epsilon}{\Longrightarrow}_{I(p)} q'$ and $p' \approxeq q'$. Lemma 15(4) implies $I(q) \subseteq I(p)$, and since $I_{\underline{A}}(r) \cap \overline{I_{\underline{A}}(p)}$ is empty, we may use Lemma 13 to obtain $q|r \stackrel{\epsilon}{\Longrightarrow}_{M} q'|r$, where $M = I(p) \cup I(r) = I(p|r)$. Thus we have $\overline{q} \stackrel{\epsilon}{\Longrightarrow}_{I(p|r)} q'|r$ and also $\langle p'|r, q'|r\rangle \in \mathcal{R}$.

(2) $r \stackrel{\tau}{\rightarrow} r'$ and $\overline{p}' \equiv p|r'$.
Since $p \approxeq q$ and p is patient, it follows that there must be a q' such that $q \stackrel{\epsilon}{\Rightarrow} q'$ and $p \approxeq q'$. We now claim that $q'|r$ is also patient. From Lemma 15(4) we may deduce $I(q') \subseteq I(p)$. Because $I_{\underline{A}}(r) \cap \overline{I_{\underline{A}}(p)} = \emptyset$, too, we have $I_{\underline{A}}(r) \cap \overline{I_{\underline{A}}(q')} = \emptyset$, whence $q'|r$ must be patient. It follows that $q'|r \stackrel{\tau}{\rightarrow} q'|r'$ and, by Lemma 12(3), we may therefore infer $q'|r \stackrel{\tau}{\Longrightarrow}_{I(q'|r)} q'|r'$. Since $I(p) = I(q')$, $I(q'|r) = I(p|r)$, and $q|r \stackrel{\epsilon}{\Rightarrow} q'|r \stackrel{\tau}{\Longrightarrow}_{I(p|r)} q'|r'$, we may conclude from Lemma 12(2) that $q|r \stackrel{\tau}{\Longrightarrow}_{I(p|r)} q'|r'$, whence $q|r \stackrel{\epsilon}{\Longrightarrow}_{I(p|r)} q'|r'$. But $\langle p|r', q'|r'\rangle \in \mathcal{R}$, as desired.

(3) $p \xrightarrow{b} p'$ and $r \xrightarrow{\overline{b}} r'$, for some $b \in A \setminus \{\tau\}$, and $\overline{p}' \equiv p'|r'$.
The proof of this case follows the same lines as the ones for the previous two cases and is therefore omitted here.

We now continue with the proof of Cond. (2). $\overline{p}\downarrow\downarrow$ means that there exists some $\overline{p}_1$ such that $\overline{p} \stackrel{\epsilon}{\Rightarrow} \overline{p}_1$ and $\overline{p}_1$ is patient. Without loss of generality, assume that if i is the least number satisfying $\overline{p} \stackrel{\tau}{\rightarrow}{}^{i} \overline{p}_1$, then $\overline{p} \stackrel{\tau}{\rightarrow}{}^{j} \overline{p}'_1$ is *not* true for any patient $\overline{p}'_1$ and $j < i$. The proof of $\overline{q}\downarrow\downarrow$ is conducted by induction upon i:

- For the base case, $i = 0$, we have that $\overline{p}$ is patient. This means that both p and r are patient and that $I_{\underline{A}}(p) \cap \overline{I_{\underline{A}}(r)} = \emptyset$. Using Lemma 15(3) we infer the existence of a patient process q_1 such that $q \stackrel{\epsilon}{\Rightarrow} q_1$ and $p \approx q_1$. Further, Lemma 15(4) implies $I(p) = I(q_1)$. Now we have that $I_{\underline{A}}(q_1) \cap \overline{I_{\underline{A}}(r)}$ is empty and that both processes q_1 and r are patient. Hence, $q_1|r$ is patient and $\overline{q} \stackrel{\epsilon}{\Rightarrow} q_1|r$. Equivalently, we have shown that $\overline{q}\downarrow\downarrow$, as desired.
- For the induction case, $i > 0$, suppose that $\overline{p} \stackrel{\tau}{\rightarrow} \overline{p}_2 \stackrel{\epsilon}{\Rightarrow} \overline{p}_1$, where $\overline{p}_1$ is patient. From Cond. (1) we may infer a process $\overline{q}_2$ such that $\overline{q} \stackrel{\epsilon}{\Rightarrow} \overline{q}_2$ and $\langle \overline{p}_2, \overline{q}_2 \rangle \in \mathcal{R}$. By induction hypothesis we have $\overline{q}_2\downarrow\downarrow$, from which $\overline{q}\downarrow\downarrow$ follows immediately. □

Because $\approx$ is insensitive to initial internal computation, it is not preserved by non-deterministic choice. This problem can be fixed in the usual fashion [25] by requiring that any silent action of one process is matched by at least one silent action of the equivalent process.

Definition 17 (*Prioritized observation congruence*).
Define $p \approx^{+} q$ to be valid if, for every $\alpha \in \mathcal{A}$, the following conditions hold:

(1) $p \xrightarrow{\alpha} p'$ implies $\begin{cases} \exists q'.\, q \stackrel{\alpha}{\Rightarrow} q' \text{ and } p' \approx q' & \text{if } \alpha \neq \tau \\ \exists q'.\, q \stackrel{\tau}{\Longrightarrow}_{I(p)} q' \text{ and } p' \approx q' & \text{if } \alpha = \tau \end{cases}$

(2) $q \xrightarrow{\alpha} q'$ implies $\begin{cases} \exists p'.\, p \stackrel{\alpha}{\Rightarrow} p' \text{ and } p' \approx q' & \text{if } \alpha \neq \tau \\ \exists p'.\, p \stackrel{\tau}{\Longrightarrow}_{I(q)} p' \text{ and } p' \approx q' & \text{if } \alpha = \tau \end{cases}$

Observe that the condition "$p\downarrow\downarrow$ if and only if $q\downarrow\downarrow$" is implicitly satisfied here. If p is patient because $p \stackrel{\tau}{\not\rightarrow}$, then $q \stackrel{\tau}{\not\rightarrow}$ due to the second condition of the above definition. Alternatively, if $p \stackrel{\tau}{\rightarrow} p'$ for some patient p', then due to the first condition of the above definition, $q \stackrel{\tau}{\Rightarrow} q'$ for some q' such that $p' \approx q'$. But then $q'\downarrow\downarrow$ by Def. 14, which in turn implies $q\downarrow\downarrow$.

To show that $\approx^{+}$ is also preserved by recursive definition, we first extend $\approx^{+}$ to open terms in the standard way [25]. Let *free*(e) denote the set of all *free* variables in expression e and let $e\{p/x\}$ denote the expression obtained from e by replacing every free occurrence of variable x by process p.

Definition 18. Let $e, f \in \mathcal{E}$ with $\mathit{free}(e) \subseteq \{x\}$ and $\mathit{free}(f) \subseteq \{x\}$. Then, $e \approx^{+} f$ if $\forall p \in \mathcal{P}.\, e\{p/x\} \approx^{+} f\{p/x\}$.

In order to prove that prioritized observation congruence $\approx^{+}$ is preserved under recursion, we need the concept of *prioritized weak bisimulation up to* along the lines of Sangiorgi and Milner [31].

Definition 19 (*Prioritized weak bisimulation up to*).
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called a *prioritized weak bisimulation up to* $\approx$ if, for all $\langle p, q \rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, the following conditions hold:

(1) $p\downarrow\downarrow$ if and only if $q\downarrow\downarrow$

$$(2)\quad p \xrightarrow{\alpha} p' \text{ implies } \begin{cases} \exists q'.\, q \stackrel{\hat{\alpha}}{\Rightarrow} q' \text{ and } p' \mathcal{R} \circ \cong q' & \text{if } \alpha \neq \tau \\ \exists q'.\, q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q' \text{ and } p' \mathcal{R} \circ \cong q' & \text{if } \alpha = \tau \end{cases}$$

$$(3)\quad q \xrightarrow{\alpha} q' \text{ implies } \begin{cases} \exists p'.\, p \stackrel{\hat{\alpha}}{\Rightarrow} p' \text{ and } p' \cong \circ\, \mathcal{R} q' & \text{if } \alpha \neq \tau \\ \exists p'.\, p \stackrel{\epsilon}{\underset{I(q)}{\Longrightarrow}} p' \text{ and } p' \cong \circ\, \mathcal{R} q' & \text{if } \alpha = \tau \end{cases}$$

The following lemma states a very useful property of "up to" relations.

Lemma 20 (Soundness of the "up to" approach)**.**
If $\mathcal{R}$ is a prioritized weak bisimulation up to $\cong$ and $\langle p, q\rangle \in \mathcal{R}$, then $p \cong q$.

Proof. We first prove that $\cong \circ\, \mathcal{R} \circ \cong$ is a prioritized weak bisimulation relation. Then the required result can be obtained in the following manner. Since $\cong$ is a reflexive relation, $p \cong p$ and $q \cong q$. Thus, if $\langle p, q\rangle \in \mathcal{R}$, we can conclude $\langle p, q\rangle \in \cong \circ\, \mathcal{R} \circ \cong$. Since relation $\cong \circ\, \mathcal{R} \circ \cong$ is proved to be a prioritized weak bisimulation relation, we further have $p \cong q$. The rest of the proof is devoted to establishing that relation $\cong \circ\, \mathcal{R} \circ \cong$ is indeed a prioritized weak bisimulation relation. This involves checking all conditions presented in Def. 14.

- Let $p \cong \circ\, \mathcal{R} \circ \cong q$. Then there exist processes p_1, q_1 such that $p \cong p_1$, $\langle p_1, q_1\rangle \in \mathcal{R}$, and $q_1 \cong q$. From the first condition of Def. 14, $p{\downarrow\downarrow}$ if and only if $p_1{\downarrow\downarrow}$ and $q_1{\downarrow\downarrow}$ if and only if $q{\downarrow\downarrow}$. Similarly, from the first condition of Def. 19, $p_1{\downarrow\downarrow}$ if and only if $q_1{\downarrow\downarrow}$. Putting these facts together we get $p{\downarrow\downarrow}$ if and only if $q{\downarrow\downarrow}$.
- Suppose $p \xrightarrow{\tau} p'$. Our aim is to prove that there is some q' such that $q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q'$ and $p' \cong \circ\, \mathcal{R} \circ \cong q'$. Since $p \cong p_1$, we know of the existence of some p_1' such that $p_1 \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} p_1'$ and $p' \cong p_1'$. Using arguments similar to those presented in the proof of Lemma 15(1), we can infer the existence of a process q_1' such that $q_1 \stackrel{\epsilon}{\underset{I(p_1)}{\Longrightarrow}} q_1'$ and $p_1' \mathcal{R} \circ \cong q_1'$. As $p \cong p_1$ and because p is patient by Lemma 15(2), we conclude $I(p_1) \subseteq I(p)$. Then, by using Lemma 12(3), we obtain $q_1 \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q_1'$. Since $q_1 \cong q$ according to Lemma 15(1), we further infer the existence of a process q' such that $q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q'$ and $q_1' \cong q'$. Because of $p' \cong p_1'$, $p_1' \mathcal{R} \circ \cong q_1'$ and $q_1' \cong q'$. By the transitivity of $\cong$ we can now conclude $p' \cong \circ\, \mathcal{R} \circ \cong q'$, as desired.

Verifying the other conditions of Def. 14 can be done in a similar fashion. □

We now have the conceptual tools necessary to prove the following compositionality result.

Theorem 21 (Compositionality of recursion)**.**
Let $e, f \in \mathcal{E}$ with $free(e) \subseteq \{x\}$ and $free(f) \subseteq \{x\}$, and suppose $e \cong^+ f$. Then, $fix(x : e) \cong^+ fix(x : f)$.

The (lengthy) proof is shown in App. A. On the basis of the above theorem and of Lemma 16, it is now easy to establish the main theorem of this section.

Theorem 22 (Congruence result)**.**
Prioritized observation congruence $\cong^+$ is a congruence with respect to all operators in CCS^{prio}. Moreover, it is the largest congruence contained in prioritized observation equivalence $\cong$.

The proof of the "largest" part of the theorem is standard and follows the lines for observation congruence and equivalence in CCS[25]. The utility of $\cong^+$ for system verification is demonstrated by means of a small example in Section 6.

4. Full-abstraction

In this subsection, we establish that prioritized observation congruence $\approx^{+}$ is a compositional variant of naïve prioritized weak bisimulation that is as insensitive to internal computation as possible. We first recall the following well-known result adapted from universal algebra.

Fact 23 (Existence of a largest congruence)**.**
Let $=$ be an equivalence over some algebra, and define $p =^{c} q$ to hold if $C[p] = C[q]$ for any context $C[]$. Then $=^{c}$ is the largest congruence contained in $=$.

Thus, we know that the largest congruence $\approx_{\mathrm{n}}^{\mathrm{c}} \subseteq \approx_{\mathrm{n}}$ contained in naïve prioritized weak bisimulation exists. The theorem we wish to establish is the following, which behaviorally characterizes this largest congruence.

Theorem 24 (Full-abstraction)**.** *$\approx_{n}^{c}$ coincides with $\approx^{+}$.*

The proof of the full-abstraction result is more difficult than the "largest" result of Thm. 22. It requires us to introduce a novel proof technique which essentially identifies $\approx_{\mathrm{n}}^{\mathrm{c}}$ by a decreasing sequence of successive approximations.

4.1. Proof technique for identifying largest congruences

Our technique for identifying largest congruences may be phrased as follows.

Proof Technique 25 (*For identifying largest congruences*)**.**
Input: An equivalence X on processes.
Output: The largest congruence contained in X.

Let $Y =_{\mathrm{df}} X$.
While Y is not a congruence do:
  Given processes p and q, choose a specific context $C(p,q)[]$ such that
    $Z =_{\mathrm{df}} \{ \langle p,q\rangle \mid \langle C(p,q)[p], C(p,q)[q]\rangle \in Y \}$ is contained in Y.
  Replace Y by an equivalence Z' with $Z \subseteq Z' \subseteq Y$.
Y is the *largest congruence* contained in X.

The validity of the technique can be established by p roving that the predicate $X^{\mathrm{c}} = Y^{\mathrm{c}}$ is an invariant of the while-loop. Clearly, the predicate is true before the first iteration, as X and Y are the same. During an iteration, Y gets replaced by an equivalence that lies in between Z and Y, and as $Y^{\mathrm{c}} = X^{\mathrm{c}} \subseteq Z$ holds by assumption and by definition of Z, respectively, we have that $X^{\mathrm{c}} \subseteq Y$ must also be true. Then the following proposition can be used to conclude that the loop invariant is maintained.

Proposition 26 (Preserving largest congruences)**.**
Let X and Y be equivalences such that $X^{c} \subseteq Y \subseteq X$. Then $X^{c} = Y^{c}$.

Proof. Since $Y \subseteq X$, it is clear that $Y^{\mathrm{c}} \subseteq X^{\mathrm{c}}$. Now we show $X^{\mathrm{c}} \subseteq Y^{\mathrm{c}}$. Let $\langle p,q\rangle \in X^{\mathrm{c}}$ and, by the hypothesis, $\langle p,q\rangle \in Y$. To prove $\langle p,q\rangle \in Y^{\mathrm{c}}$, it suffices to show that $\langle C[p], C[q]\rangle \in Y$, where $C[]$ is an arbitrary context. But X^{c} is a congruence, whence $\langle p,q\rangle \in X^{\mathrm{c}}$ implies $\langle C[p], C[q]\rangle \in X^{\mathrm{c}}$. Since $X^{\mathrm{c}} \subseteq Y$, the required result that $\langle C[p], C[q]\rangle \in Y$ follows immediately. □

If Y is a congruence, then $Y^{\mathrm{c}} = Y$. Since the while-loop is exited only when Y is a congruence, it follows that $Y = X^{\mathrm{c}}$, as desired.

Although our proof technique is presented as an algorithm, it is not guaranteed to "terminate" in the traditional sense. If one chooses contexts naively, by for example always selecting the empty context, the while-loop will not terminate. The intention behind our routine is, at every iteration of the while-loop, to obtain a relation that is congruent with respect to a larger subset of the operators in the underlying algebra. To the extent that

appropriate contexts and congruences can be defined for different operators, and that process algebras contain finitely many operators, the algorithm will nevertheless "terminate".

4.2. Applying the Proof Technique

This section applies Proof Technique 25 to proving the full-abstraction result of Thm. 24. Here, our naïve prioritized equivalence $\approx_{\mathrm{n}}$ plays the role of input X, and we proceed by iterating through the while-loop.

4.2.1. First iteration

To use our technique to identify $\approx_{\mathrm{n}}^{\mathrm{c}}$ we first introduce a specific context $\mathcal{K}_{p,q}[\,]$, given a pair of processes p and q. Intuitively, this context serves as the link between $\Rightarrow_{\mathrm{n}}$ and $\Rightarrow$. In other words, we want the $\Rightarrow_{\mathrm{n}}$-transitions of $\mathcal{K}_{p,q}[p]$ and $\mathcal{K}_{p,q}[q]$ to enable us to deduce information about the $\Rightarrow$-transitions of p and q, respectively.

The structure of context $\mathcal{K}_{p,q}[\,]$ is somewhat complex, and its description demands some auxiliary notation. Let $S_U(p)$ and $S_P(p)$ be the unprioritized sort and prioritized sort of process p, respectively, and let $S_{p,q}$ denote $S_U(p) \cup S_U(q)$. Recall that these sets are finite for any p because of the restriction we place on relabelings. We define a context $U_{p,q}[\,]$ analogous to context "$D_{p,q}[\,]$" of [10]: $U_{p,q}[r] =_{\mathrm{df}} (r[L_{p,q}])\lceil S_{p,q}$, where the relabeling function $L_{p,q}$ is

$$L_{p,q}(\alpha) =_{\mathrm{df}} \begin{cases} \alpha & \text{if } \alpha \in A \\ \alpha & \text{if } \alpha = \underline{\lambda} \text{ and } \lambda \notin S_{p,q} \\ \underline{k}_\alpha & \text{otherwise} \end{cases}$$

where $\underline{k}_\alpha \in \underline{A}$, $\underline{k}_{\overline{\alpha}} = \overline{\underline{k}_\alpha}$, $\underline{k}_\alpha \neq \underline{k}_\beta$ if $\alpha \neq \beta$, $k_\alpha \notin S_{p,q}$, and $\underline{k}_\alpha \notin S_P(p) \cup S_P(q)$. Since the set $\mathcal{A}$ of actions is infinite, such k_α are guaranteed to exist. The effect of this context $U_{p,q}[\,]$ on its "argument" is to prioritize uniquely those of its unprioritized actions which lie in $S_{p,q}$. Now we are ready to define $\mathcal{K}_{p,q}[\,]$.

Definition 27. If $L = \{l_1 \cdots l_k\} \subseteq \mathcal{A}$, then $\langle L \rangle$ represents the process $l_1.nil + \cdots + l_k.nil$, with $\langle \emptyset \rangle \equiv nil$ and $\langle \{\alpha\} \rangle \equiv \alpha.nil$. For $p, q \in \mathcal{P}$, let $\mathcal{A}_{p,q}$ denote the set $S_P(U_{p,q}[p]) \cup S_P(U_{p,q}[q])$. Also, let $\mathcal{Q}_{p,q}$ denote the process

$$fix(X : \underline{c} + \sum_{L \subseteq \overline{\mathcal{A}_{p,q}}} \underline{\tau}.(\langle L \rangle + \underline{d}_L.X)),$$

where $\underline{c} \in \underline{A} \setminus \{\underline{\tau}\}$ and $\underline{d}_L \in \underline{A} \setminus \{\underline{\tau}\}$ do not belong to $\overline{\mathcal{A}}_{p,q} \cup \mathcal{A}_{p,q}$, and $\underline{d}_L = \underline{d}_{L'}$ if and only if $L = L'$. Then, context $\mathcal{K}_{p,q}[\,]$ is defined as $\mathcal{Q}_{p,q} \mid U_{p,q}[\,]$.

The next lemma, whose proof can be found in App. B, enumerates several properties of context $\mathcal{K}_{p,q}[\,]$. Specifically if r is any process whose sort is contained in the union of the sorts of p and q, then we may use the $\Rightarrow_{\mathrm{n}}$-transitions of $\mathcal{K}_{p,q}[r]$ to infer $\Rightarrow$-transitions in r.

Lemma 28 (Properties of our context)**.**
Let r be a process whose sort is contained in the union of the sorts of processes p and q. Further, let D stand for the set of the $\underline{d}_L$-actions that occur in $\mathcal{Q}_{p,q}$.

(1) $I_{\underline{A}}(U_{p,q}[r]) = I(U_{p,q}[r])$.
(2) *Process $\mathcal{K}_{p,q}[r]$ is not patient.*
(3) *If $\mathcal{K}_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_n \mathcal{K}'$ and $\mathcal{K}' \stackrel{c}{\Rightarrow}_n$, then $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r \stackrel{\epsilon}{\Rightarrow} r'$.*
(4) *If $\mu \notin S_{p,q}, \underline{\mu} \notin D$, $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_n \mathcal{K}'$, and $\mathcal{K}' \stackrel{c}{\Rightarrow}_n$, then $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r \stackrel{\mu}{\Rightarrow} r'$.*
(5) *If $\mu \in S_{p,q}, \underline{\mu} \notin D$, $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_n \mathcal{K}'$, and $\mathcal{K}' \stackrel{c}{\Rightarrow}_n$, then $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r \stackrel{\mu}{\Rightarrow} r'$.*
(6) *If $\mathcal{K}_{p,q}[r] \stackrel{\underline{k}_\mu}{\Rightarrow}_n \mathcal{K}'$ and $\mathcal{K}' \stackrel{c}{\Rightarrow}_n$, then $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r \stackrel{\mu}{\Rightarrow} r'$.*

(7) *If* $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_n (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$, *then* $U_{p,q}[r] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r_1]$, $\overline{L} \cap M = \emptyset$, *and* $M \cap \overline{D} = \emptyset$.

(8) *If* $\mathcal{K}_{p,q}[r] \stackrel{d_L}{\Rightarrow}_n \mathcal{K}'$ *and* $\mathcal{K}' \stackrel{c}{\Rightarrow}_n$, *then* $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ *for some* r', M *such that* $U_{p,q}[r] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r'], \overline{L} \cap M = \emptyset$, *and* $M \cap \overline{D} = \emptyset$.

Given this specific context we can now define the relation

$$\approx^{\mathrm{l}}_{\mathrm{n}} =_{\mathrm{df}} \{\, \langle p, q\rangle \mid \mathcal{K}_{p,q}[p] \approx_{\mathrm{n}} \mathcal{K}_{p,q}[q] \,\},$$

which plays the role of Z in the technique. Next, we need to introduce an equivalence $\cong_{\mathrm{f}}$ which lies between $\approx^{\mathrm{l}}_{\mathrm{n}}$ and $\approx_{\mathrm{n}}$; this equivalence becomes the new value of Y in the technique. It is defined by means of a class of relations, called *finite prioritized weak bisimulations.*

Definition 29 (*Finite prioritized weak bisimulation*)**.**
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called a *finite prioritized weak bisimulation*, or fpwb in short, if, for all $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, the following conditions hold:

(1) $p \stackrel{\alpha}{\rightarrow} p'$ implies $\begin{cases} \exists q'.\, q \stackrel{\hat{\alpha}}{\Rightarrow} q' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha \neq \tau \\ \exists q'.\, q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha = \tau \end{cases}$

(2) $q \stackrel{\alpha}{\rightarrow} q'$ implies $\begin{cases} \exists p'.\, p \stackrel{\hat{\alpha}}{\Rightarrow} p' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha \neq \tau \\ \exists p'.\, p \stackrel{\epsilon}{\underset{I(q)}{\Longrightarrow}} p' \text{ and } \langle p', q'\rangle \in \mathcal{R} & \text{if } \alpha = \tau \end{cases}$

We write $p \cong_{\mathrm{f}} q$ if $\langle p, q\rangle \in \mathcal{R}$ for some finite prioritized weak bisimulation relation $\mathcal{R}$.

The only difference between a pwb and a fpwb is that pwb's require related processes to satisfy "$p\!\downarrow\!\downarrow$ if and only if $q\!\downarrow\!\downarrow$." Relation $\cong_{\mathrm{f}}$ can be shown to be an equivalence by the usual arguments. It is a congruence for divergence-free processes and thus in particular finite processes, which explains the choice of name for this behavioral relation. As can easily be verified, fpwb is a naïve weak bisimulation; it follows that $\cong_{\mathrm{f}}$ is contained in $\approx_{\mathrm{n}}$. The fact that $\approx^{\mathrm{l}}_{\mathrm{n}}$ is contained in $\cong_{\mathrm{f}}$ is obtained from the following lemma.

Lemma 30. *The equivalence* $\approx^{\mathrm{l}}_{n}$ *is a fpwb.*

Proof. Let $p \approx^{\mathrm{l}}_{\mathrm{n}} q$, i.e., $\mathcal{K}_{p,q}[p] \approx_{\mathrm{n}} \mathcal{K}_{p,q}[q]$. Suppose that $p \stackrel{\alpha}{\rightarrow} p'$. The proof splits into several cases depending upon whether $\alpha \in \underline{A}$, $\alpha \in A \setminus \{\tau\}$, or $\alpha = \tau$.

- $\alpha = \underline{\tau}$.
  Here we have $p \stackrel{\tau}{\rightarrow} p'$. We demonstrate the existence of a process q' such that $q \stackrel{\epsilon}{\Rightarrow} q'$ and $p' \approx^{\mathrm{l}}_{\mathrm{n}} q'$. As $p \stackrel{\tau}{\rightarrow} p'$, it follows that $\mathcal{K}_{p,q}[p] \stackrel{\tau}{\rightarrow} \mathcal{K}_{p,q}[p']$. From the definition of naïve weak bisimulations and the hypothesis that $\mathcal{K}_{p,q}[p] \approx_{\mathrm{n}} \mathcal{K}_{p,q}[q]$, we can infer the existence of a process $\mathcal{K}'$ such that $\mathcal{K}_{p,q}[q] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$ and $\mathcal{K}_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}'$. Since $\mathcal{K}_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}'$ and $\mathcal{K}_{p,q}[p'] \stackrel{c}{\rightarrow}$, we obtain $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$. Using Lemma 28(3) we can infer the existence of the desired process q' such that $q \stackrel{\epsilon}{\Rightarrow} q'$ and $\mathcal{K}' \equiv \mathcal{Q}_{p,q}|U_{p,q}[q']$. What we have is that $\mathcal{K}_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}_{p,q}[q']$ but our aim however is to establish $\mathcal{K}_{p',q'}[p'] \approx_{\mathrm{n}} \mathcal{K}_{p',q'}[q']$. By examining the structure of the context $\mathcal{K}_{p,q}[]$, and by using the observation that the sorts of processes p' and q' are contained in those of processes p and q, respectively, it is clear that $\mathcal{K}_{p',q'}[p'] \approx_{\mathrm{n}} \mathcal{K}_{p',q'}[q']$, i.e., $p' \approx^{\mathrm{l}}_{\mathrm{n}} q'$.
- $\alpha \in \underline{A} \setminus \{\underline{\tau}\}$ such that $\alpha = \underline{\mu}$.
  Here, the proof splits into two cases depending upon whether $\mu \in S_{p,q}$ or not. For both these cases the proof proceeds along the same lines as the one of the case $\alpha = \underline{\tau}$; however, we make use of Lemma 28(4) when $\mu \notin S_{p,q}$ in place of Lemma 28(3). On the other hand, Lemma 28(6) is utilized when $\mu \in S_{p,q}$. The case $\alpha \in A \setminus \{\tau\}$ is handled in the same fashion with the help of Lemma 28(5).

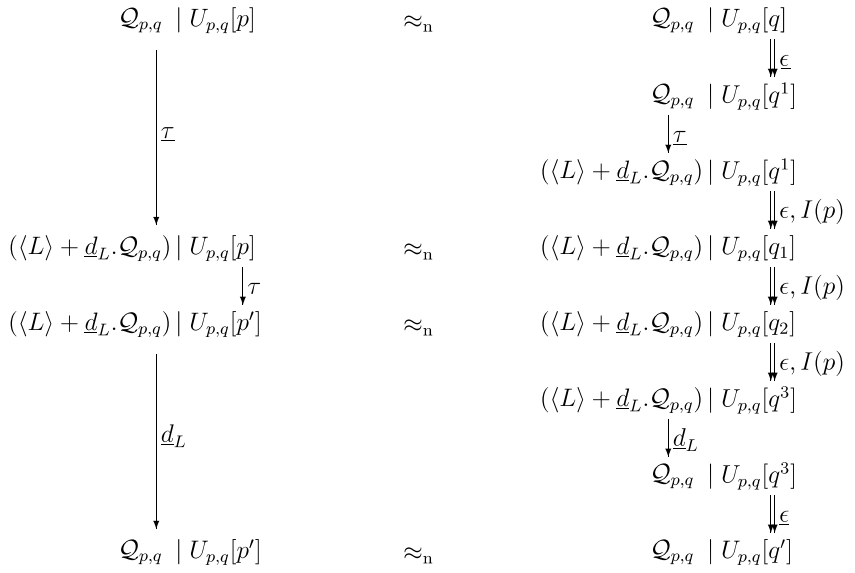

Fig. 1. Largest congruence proof: case $\alpha = \tau$.

- $\alpha = \tau$.
  We demonstrate the existence of a process q' such that $q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q'$ and $p' \approx_{\mathrm{n}}^{1} q'$. Since $\mathcal{Q}_{p,q}$ is *not* patient, transition $\mathcal{K}_{p,q}[p] \stackrel{\tau}{\rightarrow} \mathcal{K}_{p,q}[p']$ is not possible. However, the following sequence of three transitions is possible (cf. Fig. 1, left-hand side):

$$\mathcal{K}_{p,q}[p] \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[p] \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[p'] \stackrel{d_L}{\rightarrow} \mathcal{K}_{p,q}[p'],$$

  where $L =_{\mathrm{df}} \overline{\mathcal{A}_{p,q}} \setminus \overline{I(U_{p,q}[p])}$; this choice of L guarantees that process $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[p]$ is patient and makes the τ-transition possible. As $\mathcal{K}_{p,q}[p] \approx_{\mathrm{n}} \mathcal{K}_{p,q}[q]$, we can find processes $\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{K}'$ such that

$$\mathcal{K}_{p,q}[q] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}_1 \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}_2 \stackrel{\underline{d}_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}',$$

  with $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[p] \approx_{\mathrm{n}} \mathcal{K}_1$, $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}_2$, as well as $\mathcal{K}_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}'$ (cf. Fig. 1, right-hand side).
  Since $\mathcal{K}_{p,q}[p'] \approx_{\mathrm{n}} \mathcal{K}'$ and $\mathcal{K}_{p,q}[p'] \stackrel{c}{\rightarrow}$, we infer $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$. We also have $\mathcal{K}_{p,q}[q] \stackrel{\underline{d}_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. Lemma 28(8), whose validity in turn relies on Lemma 28(7), then implies $\mathcal{K}' \equiv \mathcal{K}_{p,q}[q']$ for some q', M such that $U_{p,q}[q] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[q']$, $\overline{L} \cap M = \emptyset$, and M contains no $\overline{\underline{d}}_N$. But L is $\overline{\mathcal{A}}_{p,q} \setminus \overline{I(U_{p,q}[p])}$. This means $M \subseteq I(U_{p,q}[p])$, and $U_{p,q}[q] \stackrel{\epsilon}{\underset{I(U_{p,q}[p])}{\Longrightarrow}} U_{p,q}[q']$ by Lemma 12(3). Now it is not difficult to prove that it is indeed the case that $q \stackrel{\epsilon}{\underset{I(p)}{\Longrightarrow}} q'$ and that $\mathcal{K}_{p',q'}[p'] \approx_{\mathrm{n}} \mathcal{K}_{p',q'}[q']$, i.e., $p' \approx_{\mathrm{n}}^{1} q'$. □

Summarizing we have $\approx_{\mathrm{n}}^{\mathrm{c}} \subseteq \approx_{\mathrm{n}}^{1} \subseteq \approxeq_{\mathrm{f}} \subseteq \approx_{\mathrm{n}}$. Thus, by Prop. 26, $\approx_{\mathrm{n}}^{\mathrm{c}} = \approxeq_{\mathrm{f}}^{\mathrm{c}}$.

4.2.2. Second iteration

The equivalence $\approxeq_{\mathrm{f}}$ turns out not to be congruence, as in general parallel composition is not preserved. For example, the processes in Ex. 9(4) provide an illustration of this, since it can be shown that $p \approxeq_{\mathrm{f}} q$ but $p|r \not\approxeq_{\mathrm{f}} q|r$. As $\approxeq_{\mathrm{f}}$ is not a congruence we must repeat the body of the while-loop.

Table 4
tau-axioms (left) and axioms for $\sqsubseteq_i$ (right)

$\underline{\tau}1$	$\alpha.(1.x + x) = \alpha.x$	iA1 $\alpha.x \sqsubseteq_i \alpha.y$
$\underline{\tau}2$	$\underline{\tau}.x = \underline{\tau}.x + x$	iA2 $\tau.x \sqsubseteq_i nil$
$\underline{\tau}3$	$\alpha(x + \underline{\tau}.y) = \alpha(x + \underline{\tau}.y) + \alpha.y$	iA3 $nil \sqsubseteq_i \nu.x$ $(\nu \in \mathcal{A} \setminus \{\underline{\tau}\})$
$\tau 1$	$z + \tau.(x + \tau.y) = z + \tau.(x + \tau.y) + \tau.y$ $(\vdash x \sqsubseteq_i z)$	

Our goal is now to isolate $\approx_{\mathrm{f}}^{\mathrm{c}}$, since we know by Prop. 26 that it coincides with $\approx_{\mathrm{n}}^{\mathrm{c}}$. So given a pair of processes p and q, we introduce the context $\mathcal{C}_{p,q}[\,] =_{\mathrm{df}} [\,] \mid c.nil$, where c is an unprioritized visible action that does not belong to either the sort of p or the sort of q. We further define

$$\approx_{\mathrm{n}}^{2} =_{\mathrm{df}} \{\, \langle p, q\rangle \mid \mathcal{C}_{p,q}[p] \approx_{\mathrm{f}} \mathcal{C}_{p,q}[q] \,\}.$$

This equivalence now plays the role of Z in our technique. We obviously have $\approx_{\mathrm{f}}^{\mathrm{c}} \subseteq \approx_{\mathrm{n}}^{2}$. Moreover, it can be shown that $\approx_{\mathrm{n}}^{2}$ is a pwb, which means that $\approx_{\mathrm{n}}^{2} \subseteq \approx$.

Lemma 31. *The equivalence $\approx_n^2$ is a pwb.*

Proof. To prove that $\approx_{\mathrm{n}}^{2}$ is a pwb, we show that $\approx_{\mathrm{n}}^{2}$ satisfies the conditions presented in Def. 14. It is obvious that our equivalence satisfies the second and third condition; we now focus our attention on proving the first condition. Let $p \approx_{\mathrm{n}}^{2} q$. This means that $\mathcal{C}_{p,q}[p] \approx_{\mathrm{f}} \mathcal{C}_{p,q}[q]$, where $\mathcal{C}_{p,q}[\,] \equiv [\,]|c.nil$ and neither c nor $\overline{c}$ belong to the sorts of p and q. Let $p\!\downarrow\!\downarrow$; we prove $q\!\downarrow\!\downarrow$. From the definition of the predicate $\downarrow\!\downarrow$, $p \stackrel{\epsilon}{\Rightarrow} p'$ for some patient process p'. Clearly, $p'|c.nil$ is patient. Thus, $p|c.nil \stackrel{\epsilon}{\Rightarrow} p'|c.nil \stackrel{c}{\rightarrow} p'|nil$. In other words, $p|c.nil \stackrel{c}{\Rightarrow}$. Since $p|c.nil \approx_{\mathrm{f}} q|c.nil$, we have $q|c.nil \stackrel{c}{\Rightarrow}$ as well. Consequently, $q|c.nil \stackrel{\epsilon}{\Rightarrow} q'|c.nil \stackrel{c}{\rightarrow} q'|nil$, for some patient process q'. This necessarily implies that $q \stackrel{\epsilon}{\Rightarrow} q'$, i.e., $q\!\downarrow\!\downarrow$. In a symmetric fashion, we can also establish that $q\!\downarrow\!\downarrow$ implies $p\!\downarrow\!\downarrow$. □

From the definition of $\approx_{\mathrm{f}}$, it is obvious that $\approx \subseteq \approx_{\mathrm{f}}$, and thus in our technique Y is assigned the value $\approx$. Then, from Prop. 26, it follows that $\approx_{\mathrm{f}}^{\mathrm{c}} = \approx^{\mathrm{c}}$. This concludes the second iteration of the while-loop in our proof technique for identifying largest congruences.

4.2.3. Concluding our iterations

As $\approx$ is not a congruence, we could iterate the while-loop once again. However, we instead appeal to Thm. 22, which establishes $\approx^{\mathrm{c}} = \approx^{+}$. This implies that $\approx_{\mathrm{n}}^{\mathrm{c}} = \approx_{\mathrm{f}}^{\mathrm{c}} = \approx^{\mathrm{c}} = \approx^{+}$ and concludes the proof of Thm. 24.

5. Axiomatizing prioritized observation congruence

In this section we provide a sound and complete axiomatization of prioritized observation congruence $\approx^{+}$ for *finite* processes. We achieve this by adding to the axioms of Tables 2 and 3 suitable tau-laws. As seen in the previous section, finding an appropriate set of sound tau-axioms is a non-trivial task, and their soundness may involve inclusions on initial action sets.

Let E denote the set of tau-axioms given in Table 4, on the left-hand side, plus the axioms of $E_{\simeq}$ (cf. Tables 2 and 3). Our aim is to show that E is sound and complete with respect to $\approx^{+}$. The soundness of our tau-axioms except that of Axiom $\tau 1$ is clear. Axiom $\tau 1$ involves a side condition, namely $\vdash x \sqsubseteq_i z$. The syntactic relation $\sqsubseteq_i$ is the precongruence on finite processes generated from the three axioms presented in Table 4, on the right-hand side, using the laws of inequational reasoning. The behavioral interpretation of this relation, in terms of inclusions on initial action sets, is made precise in the following lemma.

Lemma 32 (Properties of $\sqsubseteq_i$)**.**

(1) (*Soundness*) *Let $\vdash p \sqsubseteq_i q$. Then $I(p) \subseteq I(q)$, and p is patient if and only if q is patient.*
(2) (*Completeness*) *If p, q are finite processes such that $I(p) \subseteq I(q)$ and such that either both or neither is patient, then there exist some p', q' such that $p \simeq p', q \simeq q'$, and $\vdash p' \sqsubseteq_i q'$.*

The proof of the first item can be done in a straightforward manner by inducting upon the length of the proof of $\vdash p' \sqsubseteq_i q'$. The proof of the second item is somewhat more complex and can be found in App. C. The complexity mainly stems from the fact that there are no axioms like Axioms A1 through A4 of Table 2 in the axiom system presented in Table 4, right-hand side.

In the light of the above lemma, $I(x) \subseteq I(z)$ whenever $\vdash x \sqsubseteq_i z$. Hence, the soundness of Axiom $\tau 1$ is immediate. Completeness of our axioms is proved along the same lines as in [25]. Any finite process can be converted into a *normal form* by eliminating all static combinators using axioms in $E_{\simeq}$. Then we need the following analogue of Lemma 16 on p. 163 of [25].

Lemma 33 (Saturation)**.**
Let p be a normal form.

(1) $p \xrightarrow{\alpha} q$ *if and only if* $\vdash_E p = p + \alpha.q$.
(2) $p \stackrel{\alpha}{\Rightarrow} q$, *for* $\alpha \neq \tau$, *implies* $\vdash_E p = p + \alpha.q$.
(3) $p \stackrel{\tau}{\underset{I(p)}{\Longrightarrow}} q$ *implies* $\vdash_E p = p + \tau.q$.

Proof. The proofs of the first two statements proceed exactly along the same lines as those of the corresponding theorem given in [25]. We thus focus our attention on the proof of the third statement. Its proof breaks down into two cases, depending upon whether p is patient or not.

Suppose that p is not patient. Then there is some process p' such that $p \xrightarrow{\tau} p'$. By the first statement of this lemma, $\vdash_E p = p + \underline{\tau}.p'$. Axioms P and A2 imply $\vdash_E \underline{\tau}.p' = \underline{\tau}.p' + \tau.q$. Hence, $\vdash_E p = p + \underline{\tau}.p' + \tau.q = p + \tau.q$, as desired.

Now suppose that p is patient. We proceed by inducting upon the number of 1-transitions involved in $p \stackrel{\tau}{\underset{I(p)}{\Longrightarrow}} q$. For the base case we have $p \xrightarrow{\tau} q$. By the first statement of this lemma we have the required result that $\vdash_E p = p + \tau.q$. For the induction case, the proof splits into two cases:

(1) $p \stackrel{\tau}{\underset{I(p)}{\Longrightarrow}} p' \xrightarrow{\tau} q$, for some p'.
By the induction hypothesis, $\vdash_E p = p + \tau.p'$, and by the first statement of this lemma, $\vdash_E p' = p' + \underline{\tau}.q$. Putting these two results together we obtain $\vdash_E p = p + \tau.(p' + \underline{\tau}.q)$. Further, applying Axiom $\underline{\tau}3$, we get $\vdash_E p = p + \tau.(p' + \underline{\tau}.q) + \tau.q$. As shown in the previous proof step we may replace the first two terms of the right-hand side of this equation by p, whence $\vdash_E p = p + \tau.q$.
(2) $p \stackrel{\tau}{\underset{I(p)}{\Longrightarrow}} p' \xrightarrow{\tau} q$ and $I(p') \subseteq I(p)$, for some p'.
Using the induction hypothesis, $\vdash_E p = p + \tau.p'$ and the first statement of this lemma we derive $\vdash_E p = p + \tau.(p' + \tau.q)$ as in the previous case. Since it is given that $I(p') \subseteq I(p)$ and that both p and p' are patient, Axiom $\tau 1$ can be applied to the right-hand side of this equation to obtain $\vdash_E p = p + \tau.(p' + \tau.q) + \tau.q$. Again as in the previous case, the first two terms of the right-hand side of this equation can be replaced by p, and we arrive at the required result $\vdash_E p = p + \tau.q$. □

We also need an analogue of Proposition 11 on p. 156 of [25].

Proposition 34. *Let p, q be finite processes. If $p \approx q$, then $p \approx^+ q$, or $p \approx^+ q + \tau.q$, or $p \approx^+ q + \underline{\tau}.q$, or $p + \tau.p \approx^+ q$, or $p + \underline{\tau}.p \approx^+ q$.*

Proof. Let $p \approx q$ and suppose it is not the case that $p \approx^+ q$. The proof splits into three cases depending upon whether both processes are patient:

(1) Both p and q are patient.
Since p is not congruent to q we must either have $p \xrightarrow{\tau} p'$ and $p' \approx q$, for some p', or the analogous condition with the roles of p and q interchanged.

Without loss of generality we focus on the first case, for which we claim $p \cong^+ q + \tau.q$. The only non-obvious part of this statement's proof involves establishing the following condition: $q \xrightarrow{\tau} q'$ implies $\exists p_1.\ p \xRightarrow[I(q)]{\tau} p_1$ and $p_1 \cong q'$. Fix a q' such that $q \xrightarrow{\tau} q'$ and suppose the condition is not true. Then, because of the hypothesis $p \cong q$ and the fact that p is patient, $p \xRightarrow[I(q)]{\epsilon} p_2 \cong q'$ implies $p_2 \equiv p$, for any p_2. In other words, the above condition states that the *only* $\xRightarrow[I(q)]{\epsilon}$-derivative of p that is observationally equivalent to q' is p. Now we have $p \cong q$, $p' \cong q$, and $q' \cong p$, whence $p' \cong q'$ by transitivity. Since p is a finite process and $p \xrightarrow{\tau} p'$, it cannot be the case that p and p' are identical, which is a contradiction. Thus, we have $p \cong^+ q + \tau.q$.

(2) Both p and q are impatient. The proof is similar to the previous case.

(3) Exactly one of p and q is impatient.
Without loss of generality we assume p is impatient and q is patient. In this case, it is easy to see that $p \cong^+ \underline{\tau}.q$. Since q is patient, $\underline{\tau}.q \cong^+ \underline{\tau}.q + q$. Thus, by transitivity, we have the required result that $p \cong^+ \underline{\tau}.q + q$. □

We are now in a position to prove the completeness of our axiom system for finite processes.

Theorem 35 (Soundness and completeness)**.**
Let p, q be finite processes. Then $p \cong^+ q$ if and only if $\vdash_E p = q$.

Proof. The "if" part of the theorem is straightforward, so we focus on the "only if" part, i.e., on completeness. Without loss of generality, assume p and q are normal forms $\sum_{i=1}^{m} \alpha_i.p_i$ and $\sum_{j=1}^{n} \beta_j.q_j$.

The proof proceeds by induction on the sum d of the depths of p and q. For the induction base $d = 0$ we have $m = n = 0$, i.e., $p \equiv q \equiv nil$ and obviously $\vdash_E p = q$. For the induction step, let $d > 0$. It is easy to see that it is not possible for exactly one of m and n to be 0. Now we consider the case in which m, n, d are all strictly positive. It is sufficient to establish $\vdash_E p = p + q$, since $\vdash_E q = p + q$ using symmetrical arguments, which together immediately imply $\vdash_E p = q$.

To prove $\vdash_E p = p + q$ we show $\vdash_E q = \alpha_i.p_i + q$, for all $1 \le i \le m$. Choose and fix some i. The proof splits into two cases depending upon whether $\alpha_i = \tau$.

If $\alpha_i = \tau$, we have that $p \xrightarrow{\tau} p_i$ implies $q \xRightarrow[I(p)]{\tau} q'$ and $p' \cong q'$ for some q'. Obviously p is patient. Since $p \cong^+ q$, it has to be the case that q is patient as well. Lemma 15(4) then implies $I(p) = I(q)$. Thus we have $q \xRightarrow[I(q)]{\tau} q'$. By Lemma 33(33), $\vdash_E q = q + \tau.q'$, and using Prop. 34 we infer $p_i \cong^+ q'$, or $p_i \cong^+ q' + 1.q'$, or $p_i + 1.p_i \cong^+ q'$. In each of these cases, the sum of the depths of the processes involved is strictly less than the sum of the depths of the processes p and q, whence the induction hypothesis is applicable to them. Thus, one of $\vdash_E p_i = q'$, or $\vdash_E p_i + 1.p_i = q'$, or $\vdash_E p_i = q' + 1.q'$ is true. In any case $\vdash_E \tau.p_i = \tau.q'$, perhaps by using Axiom $\underline{\tau}1$. Because of $\vdash_E q = q + \tau.q'$ we can prove $\vdash_E q + \tau.p_i = q$, as desired.

If $\alpha_i \neq \tau$, then $p \xrightarrow{\alpha_i} p_i$ implies $q \overset{\alpha_i}{\Rightarrow} q'$ and $p' \cong q'$ for some process q'. By Lemma 33(33), $\vdash_E q = q + \alpha_i.q'$. Now, the rest of the proof proceeds exactly along the same lines as that of the previous case.

In summary, we have proved $\vdash_E q = \alpha_i.p_i + q$, for all $1 \le i \le m$. By summing up this equation over all i and by repeated application of Axiom A1, we obtain $\vdash_E q = p + q$. □

We conclude our axiomatization with a couple of remarks. First, Axiom (τ1) is, strictly speaking, a rule rather than an equation, which means that our axiomatization is not equational. The question naturally arises as to whether it is possible to give a finite axiomatization that is purely equational, possibly supposing that the underlying action set is finite. This issue is currently open, although we conjecture that a finite axiomatization will in fact not be possible.

Second, when extending our axiomatization to the class of *regular* processes, i.e., finite-state processes that do not contain recursion through static operators, it is not obvious how a completeness result can be obtained. The standard approach of Milner [26] relies on the possibility of removing tau-cycles in processes. In the context of global preemption, however, eliminating a $\underline{\tau}$-cycle is in general not sound, as is shown by Ex. 9(4). A similar problem has been attacked in [19] for stochastic process calculi with priority and/or maximal progress and in [8] for a different, more classical process-algebraic setting. It remains to be seen whether these techniques can be successfully applied to $\mathrm{CCS}^{\mathrm{prio}}$.

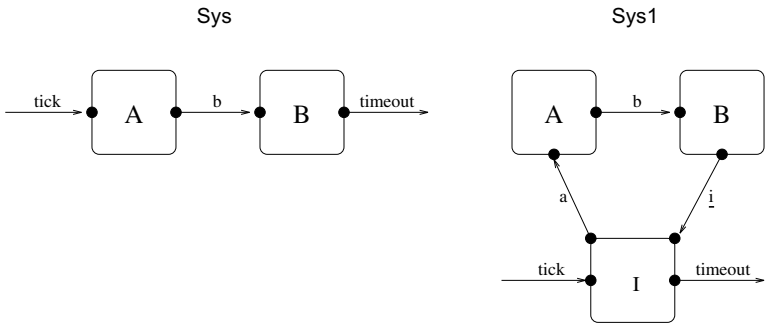

Fig. 2. Architecture of the 4-count timer implementations.

6. Example

In this section, we apply our behavioral relation of prioritized observation congruence $\approx^+$ to reason about the relationship between a specification and an implementation of a *4-count timer*, which outputs *timeout* immediately after four consecutive inputs of *tick*. A formal specification of the system may be given in CCS as follows:

$$Spec \Leftarrow tick.tick.tick.tick.\overline{timeout}.Spec\,,$$

where we use the notation $x \Leftarrow p$ instead of $fix(x : p)$.

6.1. First, naïve implementation

Suppose now that we are asked to implement *Spec* using 2-count timers, which are specified as follows:

$$Cell \Leftarrow tick.tick.\overline{timeout}.Cell\,.$$

A natural way to implement this specification is to compose two such *Cell* s serially, connecting the *timeout* port of one cell to the *tick* port of the other cell, as depicted in Fig. 2 on the left-hand side. The resulting process *Sys* is

$$Sys \Leftarrow (A|B)\backslash\{b\} \quad A \Leftarrow Cell[\overline{b}/\overline{timeout}] \quad B \Leftarrow Cell[b/tick]\,,$$

where we use the standard convention and write finite relabelings in a substitution-style notation [25].

To prove that *Sys* is correct, we can try to establish that $Sys \approx^+ Spec$. Unfortunately, this fails to be the case; in fact, the two systems are not even trace equivalent. To see this, note that *Sys* has a trace that begins with "*tick tick tick tick tick*." But in any trace of *Spec*, every four consecutive occurrences of *tick* are immediately succeeded by an occurrence of $\overline{timeout}$. Hence, *Sys* does not implement *Spec*. The cause of the problem is that the availability of a $\overline{timeout}$ action in process *B* does not preclude *A* from executing a *tick* action. The problem would persist even if we would use CCS^{prio} and prioritize $\overline{timeout}$, since visible prioritized actions do not preempt unprioritized actions.

Table 5
A prioritized weak bisimulation containing $\langle Sys3, Spec\rangle$

$\langle Sys3,$	$Spec\rangle,$
$\langle A3 \mid B3 \mid \underline{\bar{a}}.I3)\backslash M,$	$tick.tick.tick.timeout.Spec\rangle,$
$\langle \underline{a}.\underline{\bar{b}}.A3 \mid B3 \mid I3)\backslash M,$	$tick.tick.tick.timeout.Spec\rangle,$
$\langle \underline{a}.\underline{\bar{b}}.A3 \mid B3 \mid \underline{\bar{a}}.I3)\backslash M,$	$tick.tick.timeout.Spec\rangle,$
$\langle \underline{\bar{b}}.A3 \mid \underline{B3} \mid I3)\backslash M,$	$tick.tick.timeout.Spec\rangle,$
$\langle A3 \mid \underline{b}.\underline{I3}.B3 \mid I3)\backslash M,$	$tick.tick.timeout.Spec\rangle,$
$\langle A3 \mid \underline{b}.\underline{\bar{i}}.B3 \mid \underline{\bar{a}}.I3)\backslash M,$	$tick.timeout.Spec\rangle,$
$\langle \underline{a}.\underline{\bar{b}}.A3 \mid \underline{b}.\underline{\bar{i}}.B3 \mid I3)\backslash M,$	$tick.timeout.Spec\rangle,$
$\langle \underline{a}.\underline{\bar{b}}.A3 \mid \underline{b}.\underline{\bar{i}}.B3 \mid \underline{\bar{a}}.I3)\backslash M,$	$timeout.Spec\rangle,$
$\langle \underline{\bar{b}}.A3 \mid \underline{b}.\underline{\bar{i}}.B3 \mid I3)\backslash M,$	$timeout.Spec\rangle,$
$\langle A3 \mid \underline{\bar{i}}.B3 \mid I3)\backslash M,$	$timeout.Spec\rangle,$
$\langle A3 \mid B3 \mid timeout.I3)\backslash M,$	$timeout.Spec\rangle$

6.2. A still not quite correct solution

In order to develop a solution, we introduce another process, process I, that acts as a "server" for the timer. All interactions between the environment and the timer are "funneled" through this process. The resulting system, $Sys1$, is depicted on the right-hand side of Fig. 2 and is formally defined as follows:

$$Sys1 \Leftarrow (A1|B1|I)\backslash\{a,b,i\} \qquad A1 \Leftarrow A[a/tick]$$
$$I \Leftarrow tick.\bar{a}.I + \underline{i}.\overline{timeout}.I \qquad B1 \Leftarrow (B\lceil\{\overline{timeout}\})[\underline{i}/\underline{timeout}]\,.$$

However, it is still not the case that $Sys1 \cong^{+} Spec$. The reason for this inequivalence is somewhat subtle. To see why it is not possible to construct a pwb containing the pair $\langle Spec, Sys1\rangle$, suppose $Spec$ evolves to

$$Spec1 \Leftarrow tick.tick.tick.\overline{timeout}.Spec$$

by performing action $tick$. Now, there is only one process to which $Sys1$ can evolve by means of $\stackrel{tick}{\Rightarrow}$, namely $Q \equiv (A1 \mid B1 \mid \bar{a}.I)\backslash\{a,b,i\}$. So any attempt to construct a pwb relating $Spec$ and $Sys1$ must necessarily relate $Spec1$ and Q as well. The only initial action available to Q is τ; Q is incapable of a $\stackrel{tick}{\Rightarrow}$-transition. Thus, $Spec1$ and Q, and hence $Spec$ and $Sys1$, cannot be related by any pwb. In fact, the context $\mathcal{C} =_{\mathrm{df}} ([]\lceil\{tick, timeout\}) \mid (\underline{\tau} + \underline{c})$ distinguishes them: process $\mathcal{C}[Spec]$ can engage in two consecutive $tick$ actions which $\mathcal{C}[Sys1]$ cannot match. The problem is the matching of the second $tick$ action; this is because context $\mathcal{C}$ does not permit the low priority interaction between the process A and I through port a while action $\underline{c}$ is enabled.

6.3. A correct implementation

This last observation suggests that, in order to come up with a correct implementation, we should also prioritize the communications on a and b. This leads to system $Sys2$:

$$Sys2 \Leftarrow (A2 \mid B2 \mid I1)\backslash\{\underline{a},\underline{b},\underline{i}\} \qquad I1 \Leftarrow I\lceil\{a\}$$
$$A2 \Leftarrow A1\lceil\{a,b\} \qquad B2 \Leftarrow B1\lceil\{b\}\,.$$

It turns out that $Sys2$ is indeed an implementation of $Spec$, as $Sys2 \cong^{+} Spec$. To demonstrate this, one needs to build a pwb relating the two systems and then to remark that as neither system is capable of initial internal actions, equivalence with respect to $\cong$ implies equivalence with respect to $\cong^{+}$.

To ease the presentation of such a pwb we define processes $A3$, $B3$ and $I3$ that can easily be proved to be prioritized strong bisimilar to $A2$, $B2$ and $I1$, respectively; the advantage is that they do not contain any static operators:

$$Sys3 \Leftarrow (A3 \mid B3 \mid I3)\backslash\{\underline{a},\underline{b},\underline{i}\} \qquad I3 \Leftarrow tick.\underline{\bar{a}}.I3 + \underline{i}.timeout.I3$$
$$A3 \Leftarrow \underline{a}.\underline{a}.\underline{\bar{b}}.A3 \qquad B3 \Leftarrow \underline{b}.\underline{b}.\underline{\bar{i}}.B3\,.$$

Table 5, where $M =_{\mathrm{df}} \{\underline{a}, \underline{b}, \underline{i}\}$, now presents the desired pwb that contains the pair $\langle Sys3, Spec\rangle$. The proof that this relation is indeed a pwb is left to the reader.

Reflecting on this example, it must be pointed out that our 4-count timer is inherently an example of global rather than local priority [13]. This is because action *tick* is local to I and b is shared by A and B but not I.

7. Discussion and related work

This section first discusses the robustness of our approach to abstracting from internal computation in CCS$^{\mathrm{prio}}$. It then considers related work on priority in process algebras, a survey of which can be found in [13]. Our account of related work focuses largely on *CCS-based* languages with priority, and we restrict ourselves further to those languages for which observational congruences have been investigated.

7.1. Robustness of our approach

Obviously, the semantic theory of prioritized observation equivalence and congruence, and in particular our full-abstraction result, depends on the exact operators of the underlying language with priority. It is interesting to see what happens if the prioritization and deprioritization operators would be left out in our CCS$^{\mathrm{prio}}$ language. The reason for including these operators in [10] was motivated by showing $\simeq$ to be the largest congruence contained in a natural strong bisimulation induced by the CCS$^{\mathrm{prio}}$ semantics in which *all* prioritized actions, and not only action $\underline{\tau}$, preempt all unprioritized actions. It is worth noting that this result is only valid in the presence of the deprioritization operator. For example, the processes $\underline{a} + \tau.(\underline{a} + \underline{\tau})$ and $\underline{a} + \tau.\underline{\tau}$ are not related by $\simeq$, but they cannot be distinguished by any context that does not involve the deprioritization operator.

When considering CCS$^{\mathrm{prio}}$ without prioritization and deprioritization operators, it turns out that the largest congruence contained in the naïve prioritized weak bisimulation is not $\approx^{+}$ but a coarser congruence. For presenting this congruence we need to expand the definition of our weak transition relation by writing $p \overset{a}{\underset{L}{\Longrightarrow}} p'$ for $p \overset{\epsilon}{\underset{L}{\Longrightarrow}} \circ \overset{a}{\underset{L}{\longrightarrow}} \circ \overset{\epsilon}{\Rightarrow} p'$, given $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$. Note that the absence of a prioritization operator means that it is sufficient to consider prioritized initial action sets L only, since unprioritized actions may never gain preemptive power. The definition of prioritized observation equivalence may now be relaxed as follows.

Definition 36 (*"Relaxed" prioritized observation equivalence*)**.**
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *relaxed prioritized weak bisimulation relation* if, for every $\langle p, q\rangle \in \mathcal{R}$, $\underline{a} \in \underline{A}$, and $a \in A$, the following holds:

(1) $\underline{\tau} \notin I_{\underline{A}}(p)$ implies $\exists q'.\, q \overset{\epsilon}{\underset{I_{\underline{A}}(p)}{\Longrightarrow}} q', I_{\underline{A}}(q') \subseteq I_{\underline{A}}(p), \underline{\tau} \notin I_{\underline{A}}(q'), \langle p, q'\rangle \in \mathcal{R}$.
(2) $p \overset{\underline{a}}{\rightarrow} p'$ implies $\exists q'.\, q \overset{\hat{a}}{\Rightarrow} q'$, and $\langle p', q'\rangle \in \mathcal{R}$.
(3) $p \overset{a}{\rightarrow} p'$ implies $\exists q'.\, q \overset{\hat{a}}{\underset{I_{\underline{A}}(p)}{\Longrightarrow}} q'$ and $\langle p', q'\rangle \in \mathcal{R}$.
(4) $\underline{\tau} \notin I_{\underline{A}}(q)$ implies $\exists p'.\, p \overset{\epsilon}{\underset{I_{\underline{A}}(q)}{\Longrightarrow}} p', I_{\underline{A}}(p') \subseteq I_{\underline{A}}(q), \underline{\tau} \notin I_{\underline{A}}(p'), \langle p', q\rangle \in \mathcal{R}$.
(5) $q \overset{\underline{a}}{\rightarrow} q'$ implies $\exists p'.\, p \overset{\hat{a}}{\Rightarrow} p'$, and $\langle p', q'\rangle \in \mathcal{R}$.
(6) $q \overset{a}{\rightarrow} q'$ implies $\exists p'.\, p \overset{\hat{a}}{\underset{I_{\underline{A}}(q)}{\Longrightarrow}} p'$ and $\langle p', q'\rangle \in \mathcal{R}$.

We write $p \approx_{*} q$ if $\langle p, q\rangle \in \mathcal{R}$ for some relaxed prioritized weak bisimulation relation $\mathcal{R}$.

It is easy to check that each condition in the above definition relaxes the corresponding condition in Def. 14. In contrast to our earlier definition, the revised prioritized weak transition relation now allows an unprioritized a-transition to be preceded by any sequence of $\underline{\tau}$- and τ-transitions, satisfying a condition on prioritized initial action sets, and only to be trailed by $\underline{\tau}$-transitions. The corresponding notion of "relaxed" prioritized observation congruence is then given as follows.

Definition 37 (*"Relaxed" prioritized observation congruence*)**.**
Define $p \approx_*^+ q$ if, for all $\underline{a} \in \underline{A}$ and $a \in A$, the following conditions and their symmetric counterparts hold.

(1) $\forall \underline{a} \in \underline{A}.\ p \xrightarrow{\underline{a}}$ implies $q \xrightarrow{\underline{a}}$.
(2) $p \xrightarrow{\underline{a}} p'$ implies $\exists q'.\ q \stackrel{\underline{a}}{\Rightarrow} q'$ and $p' \approx_* q'$.
(3) $p \xrightarrow{a} p'$ implies $\exists q'.\ q \stackrel{a}{\Longrightarrow}_{I_{\underline{A}}(p)} q'$ and $p' \approx_* q'$.

Then $\approx_*^+$ is the largest congruence contained in $\approx_{\mathrm{n}}$; the proof of this result can be found in [23]. Since $\approx_*^+$ abstracts from more internal computations than $\approx^+$, it is sometimes more practicable for system verification. For example, if action $\underline{a}$ within process *Sys3* of our example in Sec. 6 were not prioritized, then $\underline{Sys3} \not\approx^+ Spec$. The reason is that the processes $Sys3' \Leftarrow (A3 \mid B3 \mid \overline{a}.I3)\backslash\{a, \underline{b}, \underline{i}\}$ and $Spec' \Leftarrow tick.tick.tick.timeout.Spec'$ must necessarily be prioritized weak bisimilar. However, $Sys3'$ cannot match the *tick*-transition of $Spec'$ since the former process can neither initially engage in a *tick*-transition nor in a $\underline{\tau}$-transition, but only in an unprioritized τ-transition which leads to a *tick*-transition. In contrast, $Sys3 \approx_*^+ Spec$ [23].

Another alteration of our language would be to equip $\mathrm{CCS}^{\mathrm{prio}}$ with a multi-level priority structure rather than with a two-level priority structure. Doing so is rather straightforward, given that the multi-level priority structure is defined by some complete order exhibiting a maximal element. The reason is that the main semantic concept, i.e., global preemption, does not change: any action is preempted by a synchronization on a higher prioritized port. Details of such a generalization of $\mathrm{CCS}^{\mathrm{prio}}$ can be found in [23].

7.2. Observational congruence in other languages with priority

In [7], Bol and Groote defined a weak observational congruence for a process algebra in which actions are statically assigned priorities; the priorities do not take effect, however, until a special "prioritization" process constructor is applied, which has the effect of "turning on" the priorities. Bol and Groote accomplish this by adding rules that endow processes with arbitrary look-ahead along internal transitions; this reduces weak observation congruence to a strong congruence. However, they make no attempt to establish the maximality of their congruence, and their framework would yield an undesirable semantics in our setting. For example, by following their approach it would become possible for the process $(\underline{a} + \tau.\underline{b}) \mid \overline{\underline{a}}$ to engage in a $\underline{b}$-transition, even though intuitively this should not be possible, since the prioritized internal action resulting from the synchronization of $\underline{a}$ and $\overline{\underline{a}}$ should preempt the τ-action that guards action $\underline{b}$.

Related research on extending Milner's CCS by priority has focused on adopting a *local preemption* scheme, rather than a global preemption scheme [9,12]. Whereas Camilleri, Winskel and Jensen introduced to CCS a *prioritized choice* operator and a *prioritized parallelism* operator in the style of *occam*'s *prialt* and *pripar* constructs [9,21], the present authors started from the $\mathrm{CCS}^{\mathrm{prio}}$ language of Section 2.1, with the prioritization and deprioritization operators removed [12]. Both approaches have been shown to be roughly equivalent and have essentially identified the same observation congruence [12,21]. While the former approach presents a complete axiomatization of this observation congruence for finite processes, the latter adds a full-abstraction result whose proof re-used the proof technique of Section 4.

Phillips recently presented another approach to extending CCS by priority [29], which is inspired by Camilleri and Winskel's account but does not require one to treat input and output actions asymmetrically. Priority is introduced to non-deterministic choice: $\sum_i S_i{:}a_i.p_i$ allows action a_i to be executed only when the environment refuses synchronizations on all actions in the priority guard S_i. The adopted semantics is somewhat surprising and non-standard in that a process can offer a synchronization on a particular port while this synchronization is preempted at the same time. The chosen adaptation of observation congruence interestingly differs from the one in CCS in that a weak a-transition may only engage in internal computation before executing action a and not also afterwards. However, it turns out to be a congruence and enables an elegant axiomatization, too.

Last, but not least, Prasad extended his *Calculus of Broadcasting Systems* to include a notion of static priority and global preemption [30]. He also investigated a semantic theory based on Milner's observation equivalence

[25]. Remarkably, this theory does not suffer from the technical subtleties experienced in CCS^{prio} since his calculus uses a much simpler model for communication which is based on the principle of *broadcasting*. In this setting, priority values are only attached to output actions which cannot be restricted or hidden as in traditional process algebras. Finally, it should be mentioned that Prasad's calculus contains an operator, called *translate*, which enables the prioritization and the deprioritization of actions.

8. Conclusions and future work

In this article, we have investigated the problem of defining a behavioral equivalence for Cleaveland and Hennessy's process algebra CCS^{prio} with priority that relates processes on the basis of their observable behavior. Taking the CCS^{prio} framework of [10] as the foundation, we first extended a strong bisimulation congruence by naïvely abstracting from *all* internal computation and thus obtained a naïve observation equivalence. Unfortunately, this relation does not satisfy Milner's tau-laws [25] and is indeed not a congruence because of its insensitivity to the potential preemptability of transitions. We defined prioritized observation congruence to retain such sensitivity and showed it to be the largest such relation contained in the naïve observation equivalence. Establishing the largest-congruence result needed more ingenuity than typically required to prove a similar theorem in other process algebras, owing to the fact that the base relation, i.e., the naïve observation equivalence, was not preserved under parallel composition. Indeed, to complete the task we had to introduce a new proof technique that relies on successive approximations towards the desired largest congruence. This proof technique has recently been re-used to solve similar full-abstraction problems in other CCS-based process algebras with preemption [11,12,24] and as such deserves to be documented in the literature. We also presented an algebraic characterization of our prioritized observation congruence for finite processes and gave an example illustrating the utility of this congruence for system verification.

Regarding future work, the adaptation of other observation equivalences for processes with priorities should be investigated. In particular, adapting *branching bisimulation equivalence* [17] promises to yield a somewhat simpler congruence than the observation congruence of this article. Another line of research is to develop and implement efficient algorithms to compute prioritized weak bisimulations, as outlined in App. D.

Acknowledgments

We thank Ivan Christoff and Steve Sims for several useful discussions, as well as the anonymous referees for their constructive comments and suggestions.

Appendix A. Proof: compositionality of prioritized observation congruence for recursion

In this appendix we prove Thm. 21. Let $e, f \in \mathcal{E}$ with $\mathit{free}(e) \subseteq \{x\}$ and $\mathit{free}(f) \subseteq \{x\}$, and suppose $e \approx^{+} f$. We need to establish that $\mathit{fix}(x : e) \approx^{+} \mathit{fix}(x : f)$. To do so, let $p =_{\text{df}} \mathit{fix}(x : e)$ and $q =_{\text{df}} \mathit{fix}(x : f)$. Further, let $\mathcal{R}$ be the relation

$$\{\, \langle g\{p/x\}, g\{q/x\}\rangle \mid g \in \mathcal{E}, \mathit{free}(g) \subseteq \{x\} \,\}\,.$$

Henceforth, we write $g(p)$ for $g\{p/x\}$ and $g(q)$ for $g\{q/x\}$.

First we show that, for any $\langle g(p), g(q)\rangle \in \mathcal{R}$, the following conditions hold:

(1) $g(p)\!\downarrow\!\downarrow$ implies $g(q)\!\downarrow\!\downarrow$

(2) $g(p) \xrightarrow{\alpha} \overline{p}$ implies $\begin{cases} \exists \overline{q}.\, g(q) \stackrel{\alpha}{\Rightarrow} \overline{q} \text{ and } \overline{p}\,\mathcal{R} \circ \approx \overline{q} & \text{if } \alpha \neq \tau \\ \exists \overline{q}.\, g(q) \stackrel{\tau}{\underset{I(g(p))}{\Longrightarrow}} \overline{q} \text{ and } \overline{p}\,\mathcal{R} \circ \approx \overline{q} & \text{if } \alpha = \tau \end{cases}$

(3) $g(q) \stackrel{\alpha}{\rightarrow} \overline{q}$ implies $\begin{cases} \exists \overline{p}.\, g(p) \stackrel{\alpha}{\Rightarrow} \overline{p} \text{ and } \overline{p} \cong \circ\, \mathcal{R}\, \overline{q} & \text{if } \alpha \neq \tau \\ \exists \overline{p}.\, g(p) \stackrel{\tau}{\Longrightarrow}_{I(g(q))} \overline{p} \text{ and } \overline{p} \cong \circ\, \mathcal{R}\, \overline{q} & \text{if } \alpha = \tau \end{cases}$

Second, we show that $\mathcal{R}$ is a prioritized weak bisimulation up to $\cong$; this follows immediately from the proof of the conditions stated above. By taking $g \equiv x$, we have $\langle p, q\rangle \in \mathcal{R}$ and hence $p \cong q$ with the help of Lemma 20. An examination of the above conditions shows that in fact $p \cong^{+} q$ is true, which is the required result. The rest of the proof is devoted to establishing the above mentioned conditions.

Conds. (2) and (3), for $\alpha \neq \tau$, can be proved exactly in the same way as the corresponding results for CCS[25]. Now we establish the "only if" part of Cond. (1); in fact, we prove the following stronger lemma.

Lemma 38. *$g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ and $\overline{p}$ is patient implies the existence of some $\overline{q}$ such that $g(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}$, $\overline{q}$ is patient, and $I(\overline{p}) = I(\overline{q})$.*

Proof. This result is proved by inducting upon the depth of inference of the transition $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$. The proof splits into nine cases depending upon the structure of g:

- $g \equiv x$, i.e., $g\{p/x\} \equiv p$.
  Thus $p \stackrel{\epsilon}{\Rightarrow} \overline{p}$ must have been inferred from $e\{p/x\} \stackrel{\epsilon}{\Rightarrow} \overline{p}$, and the latter transition has a shorter depth of inference. Hence, one may apply the induction hypothesis to conclude the existence of a patient process $\overline{q}'$ such that $e\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}'$ and $I(\overline{p}) = I(\overline{q}')$. Since it is known that $e \cong^{+} f$, we have $e(q) \cong^{+} f(q)$ by Def. 18.
  Using Lemma 15(5), there is a process $\overline{q}$ satisfying $f(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}$, $\overline{q}$ is patient, and $\overline{q} \cong \overline{q}'$. Then, Lemma 15(4) implies $I(\overline{q}') = I(\overline{q})$. Since $q \equiv fix(x : f)$, it follows from the operational semantics of the "fix-point" constructor that $q \stackrel{\epsilon}{\Rightarrow} \overline{q}$ and $I(\overline{p}) = I(\overline{q})$.
- $g \equiv \alpha.g_1$.
  If $\alpha \neq \underline{\tau}$, then both $g(p)$ and $g(q)$ are patient and $I(g(p)) = I(g(q))$; the required result is immediate. Let us assume $\alpha = \underline{\tau}$. Then $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ must have been inferred from $g_1(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$, whence the latter transition has a shorter depth of inference. So one may apply the induction hypothesis to infer the existence of a patient process $\overline{q}$ such that $g_1\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}$ and $I(\overline{p}) = I(\overline{q})$. Then, it is obvious that $g(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}$, as we are assuming $g \equiv \underline{\tau}.g_1$.
- $g \equiv g_1 + g_2$.
  The transition $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ could have been inferred from $g_1(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ or $g_2(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$. W.l.o.g., we assume it was inferred from transition $g_1(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ which has a shorter depth of inference. So one may apply the induction hypothesis to conclude the existence of a patient process $\overline{q}$ such that $g_1\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}$ and $I(\overline{p}) = I(\overline{q})$. Then, $g(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}$ since we are in the case of $g \equiv g_1 + g_2$.
- $g \equiv g_1 | g_2$.
  $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ implies that $\overline{p}$ is of the form $\overline{p}_1 | \overline{p}_2$. Since $\overline{p}$ is patient, so are $\overline{p}_1$ and $\overline{p}_2$. Further, the initial action set $I_{\underline{A}}(\overline{p}_1) \cap \overline{I_{\underline{A}}(\overline{p}_2)}$ is empty. The proof now splits into two cases depending on how the transition $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ could have been inferred:
  (1) It could have been inferred from the transitions $g_1(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}_1$ and $g_2(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}_2$. So one may apply the induction hypothesis to conclude the existence of patient processes $\overline{q}_1$ and $\overline{q}_2$ such that $g_1\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}_1$ and $I(\overline{p}_1) = I(\overline{q}_1)$, and $g_2\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}_2$ and $I(\overline{p}_2) = I(\overline{q}_2)$, respectively. Since $I(\overline{p}_1) \cap \overline{I(\overline{p}_2)} \cap \underline{A}$ is empty, it is also the case that $I(\overline{q}_1) \cap \overline{I(\overline{q}_2)} \cap \underline{A}$ is empty. Thus, $\overline{q}_1 | \overline{q}_2$ is patient and $I(\overline{q}_1 | \overline{q}_2) = I(\overline{p})$.
  (2) There exists some $s \in (\underline{A} \setminus \{\underline{\tau}\})^{+}$ such that $g_1(p) \stackrel{\bar{s}}{\Rightarrow} \overline{p}_1$ and $g_2(p) \stackrel{\bar{s}}{\Rightarrow} \overline{p}_2$. Applying Cond. (2) repeatedly we infer the existence of processes $\overline{q}_1$ and $\overline{q}_2$ such that $g_1(q) \stackrel{s}{\Rightarrow} \overline{q}_1$, $g_2(q) \stackrel{\bar{s}}{\Rightarrow} \overline{q}_2$, $\overline{p}_1 \mathcal{R} \circ \cong \overline{q}_1$, and $\overline{p}_2 \mathcal{R} \circ \cong \overline{q}_2$. Clearly, there exists a process $\overline{r}_1$ such that $\overline{p}_1 \mathcal{R} \overline{r}_1 \cong \overline{q}_1$. Since the transition $\overline{p}_1 \stackrel{\epsilon}{\Rightarrow} \overline{p}_1$ has a shorter depth of inference and since $\overline{p}_1$ is patient, the induction hypothesis can be applied to conclude the existence of a patient process r_1' such that $r_1 \stackrel{\epsilon}{\Rightarrow} r_1'$ and $I(\overline{p}_1) = I(r_1')$. By using Lemma 15(4) we again infer the existence of a patient process $\overline{q}_1'$ such that $\overline{q}_1 \stackrel{\epsilon}{\Rightarrow} \overline{q}_1'$ and $I(\overline{q}_1') = I(r_1')$. Putting these facts together we have shown that $g_1(q) \stackrel{s}{\Rightarrow} \overline{q}_1'$ and that $\overline{q}_1'$ is patient with $I(\overline{q}_1') = I(\overline{p}_1)$. Arguing in a similar fashion we can prove the

existence of a process $\overline{q}'_2$ such that $g_2(q) \stackrel{\bar{s}}{\Rightarrow} \overline{q}'_2$ and $\overline{q}'_2$ is patient with $I(\overline{q}'_2) = I(\overline{p}_2)$. Thus, $\overline{q}'_1|\overline{q}'_2$ is patient with $g_1(q)|g_2(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}'_1|\overline{q}'_2$ and $I(\overline{q}'_1|\overline{q}'_2) = I(\overline{p})$.

- $g \equiv g_1[S]$, or $g_1\lceil\mu$, or $g_1\lfloor\underline{\mu}$, or $g_1\backslash\lambda$.
  In all these cases, the transition $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ must have been inferred from transition $g_1(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$, so that the induction hypothesis is applicable. Hence, there exists a patient process $\overline{q}$ such that $g_1(q) \stackrel{\epsilon}{\Rightarrow} \overline{q}$ and $I(\overline{q}) = I(\overline{p})$.
- $g \equiv fix(y : h)$.
  $g(p) \stackrel{\epsilon}{\Rightarrow} \overline{p}$ means that $(fix(y : h))\{p/x\} \stackrel{\epsilon}{\Rightarrow} \overline{p}$. This transition must have been inferred from $(h\{fix(y : h)/y\})\{p/x\} \stackrel{\epsilon}{\Rightarrow} \overline{p}$, which has a shorter proof. So we can apply induction hypothesis to obtain the existence of a patient process $\overline{q}$ with $I(\overline{q}) = I(\overline{p})$ and $(h\{fix(y : h)/y\})\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}$. Using the operational semantics of the "fix-point" constructor we can now conclude $(fix(y : h))\{q/x\} \stackrel{\epsilon}{\Rightarrow} \overline{q}$. □

It is easy to see that the above lemma proves the "only if" part of Cond. (1); the "if" part is proved in a symmetric manner. To prove Cond. (2) for $\alpha = \tau$, the following lemma is useful.

Lemma 39. *If $g(p)$ is patient, then $g(q)$ is patient and $I(g(q)) = I(g(p))$.*

Proof. If $g(q)$ is not patient, then, using Cond. (3) with $\alpha = \underline{\tau}$, we infer $g(p) \stackrel{\tau}{\Rightarrow}$, which is a contradiction to the assumption that $g(p)$ is patient. Since both $g(p)$ and $g(q)$ are patient, using arguments similar to those presented in the proof of Lemma 15(4), we conclude $I(g(q)) = I(g(p))$. □

Now we turn our attention to the proof of Cond. (2) for $\alpha = \tau$.

Lemma 40. *$g(p) \stackrel{\tau}{\rightarrow} \overline{p}$ implies the existence of some $\overline{q}$ such that $g(q) \underset{I(g(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}$ and $\overline{p}\,\mathcal{R} \circ \approx \overline{q}$.*

Proof. This result is proved by inducting upon the depth of inference of the transition $g(p) \stackrel{\tau}{\rightarrow} \overline{p}$. The proof splits into nine cases depending upon the structure of g.

- $g \equiv x$, i.e., $g\{p/x\} \equiv p$.
  Thus, $p \stackrel{\tau}{\rightarrow} \overline{p}$ must have been inferred from $e\{p/x\} \stackrel{\tau}{\rightarrow} \overline{p}$ and, therefore, the latter transition has a shorter depth of inference. So one may apply the induction hypothesis to obtain the existence of a process $\overline{q}'$ such that $e\{q/x\} \underset{I(e(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}'$ and $\overline{p}\,\mathcal{R} \circ \approx \overline{q}'$. Lemma 39 allows us to conclude that $e(q)$ is patient and that $I(e(q)) = I(e(p))$, as we are given $e(p) \stackrel{\tau}{\rightarrow}$. Since we know $e \approx^+ f$, we have $e(q) \approx^+ f(q)$ by Def. 18. Now it is possible to show—using induction upon the number of 1-transitions involved in transition $e\{q/x\} \underset{I(e(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}'$ and observing that $I(e(q)) = I(e(p))$—that $f(q) \underset{I(e(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}$ and $\overline{q}' \approx \overline{q}$, for some process $\overline{q}$. From the operational semantics of the "fix-point" constructor we conclude $q \underset{I(e(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}$ and $\overline{q}' \approx \overline{q}$. But we already have $\overline{p}\,\mathcal{R} \circ \approx \overline{q}'$. By transitivity we get $\overline{p}\,\mathcal{R} \circ \approx \overline{q}$, as desired.
- $g \equiv \alpha.g_1$.
  If $\alpha \neq \tau$, the required result is vacuously true. Let us assume $\alpha = \tau$, whence $g(p) \stackrel{\tau}{\rightarrow} \overline{p}$ implies $\overline{p} \equiv g_1(p)$. Then, it is clear that $g(q) \equiv \tau.g_1(q) \underset{I(g(q))}{\stackrel{\tau}{\Longrightarrow}} g_1(q)$ and $g_1(p)\,\mathcal{R} \circ \approx g_1(q)$. By Lemma 39 we conclude $I(g(p)) = I(g(q))$. The required result is then immediate.
- $g \equiv g_1|g_2$.
  If $g(p) \stackrel{\tau}{\rightarrow} \overline{p}$, then $\overline{p}$ is of the form $\overline{p}_1|\overline{p}_2$. Further, both $g_1(p)$ and $g_2(p)$ are patient, with $I_{\underline{A}}(g_1(p)) \cap \overline{I_{\underline{A}}(g_2(p))}$ being empty. The proof now splits into three cases depending on the transition from which $g(p) \stackrel{\tau}{\rightarrow} \overline{p}$ was inferred:
  (1) $g_1(p) \stackrel{\tau}{\rightarrow} \overline{p}_1$ and $g_2(p) \equiv \overline{p}_2$.
  Applying the induction hypothesis to the transition $g_1(p) \stackrel{\tau}{\rightarrow} \overline{p}_1$ we know of the existence of some $\overline{q}_1$ such that $g_1(q) \underset{I(g_1(p))}{\stackrel{\tau}{\Longrightarrow}} \overline{q}_1$ and $\overline{p}_1\,\mathcal{R} \circ \approx \overline{q}_1$. As $g_2(p)$ is patient, we can use Lemma 39 to conclude that $g_2(q)$ is

patient, too, and that $I(g_2(q)) = I(g_2(p))$. By Lemma 13 we then deduce $g_1(q)|g_2(q) \stackrel{\tau}{\underset{I(g(p))}{\Longrightarrow}} \overline{q}_1|g_2(q)$. Now we like to prove $\overline{p}_1|g_2(p)\,\mathcal{R}\circ\cong\overline{q}_1|g_2(q)$.

We have already proved $\overline{p}_1\,\mathcal{R}\circ\cong\overline{q}_1$. Then there exists some process r such that $\overline{p}_1\,\mathcal{R}\,r\cong\overline{q}_1$. From the definition of $\mathcal{R}$ we conclude $\overline{p}_1 \equiv h(p)$ and $h(q) \equiv r$, for some $h \in \mathcal{E}$. Since $\cong$ is preserved under parallel composition, $h(q)\cong\overline{q}_1$ implies $h(q)|g_2(q)\cong\overline{q}_1|g_2(q)$. Again, using the definition of $\mathcal{R}$, we obtain $\langle h(p)|g_2(p), h(q)|g_2(q)\rangle \in \mathcal{R}$. Since $\overline{p}_1 \equiv h(p)$ we have proved $\overline{p}_1|g_2(p)\,\mathcal{R}\circ\cong\overline{q}_1|g_2(q)$.

(2) $g_2(p) \stackrel{\tau}{\rightarrow} \overline{p}_2$ and $g_1(p) \equiv \overline{p}_1$.
The proof of this case is symmetrical to that of the previous one.

(3) $g_1(p) \stackrel{\mu}{\rightarrow} \overline{p}_1$ and $g_2(p) \stackrel{\overline{\mu}}{\rightarrow} \overline{p}_2$.
The proof of this case also uses arguments similar to those presented for the first case.

- The proofs of the cases in which g takes on other forms do not require any new proof technique and are thus omitted here. □

Cond. (3) is proved using arguments similar to those presented in the previous lemma. This concludes the proof of Thm. 21.

Appendix B. Full-abstraction result: auxiliary statements for its proof

This section proves the statements of Lemma 28.

(1) Obvious.

(2) $\mathcal{K}_{p,q}[r] \equiv \mathcal{Q}_{p,q}|U_{p,q}[r]$ and $\mathcal{Q}_{p,q} \stackrel{\tau}{\not\rightarrow}$. Hence, process $\mathcal{K}_{p,q}[r]$ is not patient.

(3) The proof proceeds by induction upon the number of 1-transitions involved in $\mathcal{K}_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. The base case is trivial, by taking $r' \equiv r$. For the induction step, we have $\mathcal{K}_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'' \stackrel{1}{\rightarrow} \mathcal{K}'$. Since $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$, it is also the case that $\mathcal{K}'' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$. Then the induction hypothesis is applicable to transition $\mathcal{K}_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}''$, whence $\mathcal{K}'' \equiv \mathcal{K}_{p,q}[r'']$ for some r'' such that $r \stackrel{\epsilon}{\Rightarrow} r''$. As $\mathcal{K}_{p,q}[r'']$ is not patient according to Lemma 28(2), transition $\mathcal{K}'' \stackrel{1}{\rightarrow} \mathcal{K}'$ has to be of the form $\mathcal{K}_{p,q}[r''] \stackrel{\tau}{\rightarrow} \mathcal{K}'$. Note that any $\underline{\tau}$-derivative of $\mathcal{K}_{p,q}[r'']$ has to be of the form $\mathcal{K}_{p,q}[r']$, for some process r', or of the form $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r'']$.
If $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$, then it is obvious that $r'' \stackrel{\tau}{\rightarrow} r'$, and we are done. Now suppose that $\mathcal{K}' \equiv (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r'']$. In this case, $\mathcal{K}'$ cannot engage in a $\stackrel{c}{\Rightarrow}_{\mathrm{n}}$-transition. To see this, assume $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r''] \stackrel{c}{\Rightarrow}_{\mathrm{n}}$ to be true. The only way this is possible is through the execution of a synchronization over $\underline{d}_L$, which requires $U_{p,q}[r''] \stackrel{\overline{d}_L}{\Rightarrow}_{\mathrm{n}}$. But the latter is not true because $\underline{d}_L \notin \overline{\mathcal{A}}_{p,q}$. Thus we have shown that $\mathcal{K}'$ cannot engage in a $\stackrel{c}{\Rightarrow}_{\mathrm{n}}$-transition. This concludes the proof of the induction step.

(4) The proof is done by induction upon the number of transitions involved in $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. For the base case we have $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\rightarrow} \mathcal{K}'$. Since $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$, it cannot be the case that $\underline{\mu} \equiv \underline{c}$. Consequently, $\mathcal{K}'$ has to be of the form $\mathcal{Q}_{p,q}|U_{p,q}[r']$, for some r' such that $U_{p,q}[r] \stackrel{\mu}{\rightarrow} U_{p,q}[r']$. But $U_{p,q}[] \equiv ([][L_{p,q}])\lceil S_{p,q}$. Since $\mu \notin S_{p,q}$, the transition $(r[L_{p,q}])\lceil S_{p,q} \stackrel{\mu}{\rightarrow} (r'[L_{p,q}])\lceil S_{p,q}$ implies that $r[L_{p,q}] \stackrel{\mu}{\rightarrow} r'[L_{p,q}]$; the $\underline{\mu}$-transition could not have arisen from the prioritization of a μ-transition. Since the relabeling function $\overline{L}_{p,q}$ acts as an identity function on actions such as $\underline{\mu}$, for which $\mu \notin S_{p,q}$, transition $r[L_{p,q}] \stackrel{\underline{\mu}}{\rightarrow} r'[L_{p,q}]$ implies that $r \stackrel{\underline{\mu}}{\rightarrow} r'$. This is the required result for the base case. For the induction step, the proof splits into two cases.

(a) $\mathcal{K}_{p,q}[r] \stackrel{\tau}{\rightarrow} \mathcal{K}'' \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$.
If $\mathcal{K}'' \equiv \mathcal{K}_{p,q}[r'']$ for some r'', then it is easy to see that $r \stackrel{\tau}{\Rightarrow} r''$. On the other hand, if $\mathcal{K}'' \equiv (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r]$, we can prove—using arguments similar to those used in the proof of the previous statement—that $\mathcal{K}'$ cannot engage in a $\stackrel{c}{\Rightarrow}_{\mathrm{n}}$-transition, which is a contradiction. Consequently, $\mathcal{K}''$ has to be of the form $\mathcal{K}_{p,q}[r'']$, for some r'' with $r \stackrel{\tau}{\Rightarrow} r''$.

Now we have $\mathcal{K}_{p,q}[r''] \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. We apply the induction hypothesis to this transition to infer the existence of a process r' such that $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ and $r'' \stackrel{\mu}{\Rightarrow} r'$. Hence, $r \stackrel{\epsilon}{\Rightarrow} r''$, in addition to $r'' \stackrel{\mu}{\Rightarrow} r'$. Thus, the required result that $r \stackrel{\mu}{\Rightarrow} r'$ and $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ is immediate.

(b) $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'' \stackrel{1}{\rightarrow} \mathcal{K}'$.
Since $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$ and $\mathcal{K}'' \stackrel{1}{\rightarrow} \mathcal{K}'$, we have $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$. Now the proof proceeds similar to the previous case, by applying the induction hypothesis to the first transition $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}''$ and Lemma 28(3) to the second transition $\mathcal{K}'' \stackrel{1}{\rightarrow} \mathcal{K}'$.

This concludes the proof of the induction step and also of Lemma 28(4).

(5) The proof is done by induction upon the number of transitions involved in $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. For the base case we have $\mathcal{K}_{p,q}[r] \stackrel{\mu}{\rightarrow} \mathcal{K}'$. Since $\mathcal{K}' \stackrel{c}{\Rightarrow}_{\mathrm{n}}$, it cannot be the case that $\underline{\mu} \equiv \underline{c}$. Consequently, $\mathcal{K}'$ has to be of the form $\mathcal{Q}_{p,q}|U_{p,q}[r']$, for some r' such that $U_{p,q}[r] \stackrel{\mu}{\rightarrow} U_{p,q}[r']$. But $U_{p,q}[] \equiv ([][L_{p,q}])\lceil S_{p,q}$, which gives rise to two possibilities: $r[L_{p,q}] \stackrel{\mu}{\rightarrow} r'[L_{p,q}]$ or $r[L_{p,q}] \stackrel{\mu}{\rightarrow} r'[L_{p,q}]$. Suppose the former holds. This implies $r \stackrel{\alpha}{\rightarrow} r'$, where $L_{p,q}(\alpha) = \underline{\mu}$. By examining the definition of the relabeling function $L_{p,q}$, it is easy to see that α is $\underline{\mu}$. But we are given $\overline{\mu} \in S_{p,q}$. Then process $r[L_{p,q}]$ cannot engage in a $\underline{\mu}$-transition, which is a contradiction. So the latter possibility has to be true, for which it is easy to see that $r \stackrel{\mu}{\rightarrow} r'$; this is the required result for the base case. The proof of the induction step is similar to that of Lemma 28(4).

(6) The proof is similar to that of Lemma 28(4).

(7) The result is again proved by induction upon the number of 1-transitions involved in $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$. The base case is trivially true. For the induction step there are two cases to consider:

(a) $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} r^* \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$.
We claim that r^* must be of the form $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*]$, for some process r_*. Equivalently, we claim that process r^* cannot be of the form $\mathcal{Q}_{p,q}|U_{p,q}[r_1]$. To see why this claim is true observe that, in order for the transition $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{Q}_{p,q}|U_{p,q}[r_1]$ to be possible, it is necessary for a synchronization over $\underline{d}_L$ to take place. However, since $U_{p,q}[r]$ is not capable of a $\stackrel{\overline{d}_L}{\Rightarrow}_{\mathrm{n}}$-transition, such a synchronization is not possible, whence our claim is true.
Now, we have $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*] \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$. We apply the induction hypothesis to transition $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*]$ in order to infer $U_{p,q}[r] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r_*]$, for some M such that $\overline{L} \cap M = \emptyset$. Considering the transition $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*] \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$, it is obvious that $U_{p,q}[r_*] \stackrel{\tau}{\rightarrow} U_{p,q}[r_1]$. Thus, we have shown the required result that $U_{p,q}[r] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r_1]$ for some M such that $\overline{L} \cap M = \emptyset$.

(b) $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} r^* \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$.
Again, it is not difficult to see that process r^* has to be of the form $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*]$, for some process r_*. As before, we apply the induction hypothesis to transition $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \stackrel{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*]$ to infer $U_{p,q}[r] \stackrel{\epsilon}{\underset{M_1}{\Longrightarrow}} U_{p,q}[r_*]$, for some M_1 such that $\overline{L} \cap M_1 = \emptyset$. From transition $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*] \stackrel{\tau}{\rightarrow} (\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$, it is obvious that $U_{p,q}[r_*] \stackrel{\tau}{\rightarrow} U_{p,q}[r_1]$.
As $(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_*]$ is patient, we have $\overline{I_{\underline{A}}(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q})} \cap I_{\underline{A}}(U_{p,q}[r_1]) = \emptyset$. But $I(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q}) = L \cup \{\underline{d}_L\} = \overline{I_{\underline{A}}(\langle L\rangle + \underline{d}_L.\mathcal{Q}_{p,q})}$. Take $M = M_1 \cup I(U_{p,q}[r_1])$. This choice of M satisfies the conditions $\overline{L} \cap M = \emptyset$ and $U_{p,q}[r_*] \stackrel{\tau}{\underset{M}{\Longrightarrow}} U_{p,q}[r_1]$, (cf. Lemma 28(1)). The required result $U_{p,q}[r] \stackrel{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r_1]$ then follows easily with the help of Lemma 12(3) and the transitivity of $\stackrel{\epsilon}{\underset{M}{\Longrightarrow}}$.

(8) The proof is done by induction upon the number of transitions involved in $\mathcal{K}_{p,q}[r] \stackrel{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. The base case is vacuously true because $\mathcal{K}_{p,q}[r]$ cannot engage in a $\underline{d}_L$-transition. For the induction step, the proof splits into two cases:

(a) $\mathcal{K}_{p,q}[r] \xrightarrow{\tau} \mathcal{K}'' \overset{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$.
The proof splits into further two cases depending upon the structure of $\mathcal{K}''$.
(i) $\mathcal{K}'' \equiv \mathcal{K}_{p,q}[r'']$ for some r''.
It is obvious that $r \xrightarrow{\tau} r''$, and therefore $U_{p,q}[r] \xrightarrow{\tau} U_{p,q}[r'']$. Now we may apply the induction hypothesis to $\mathcal{K}_{p,q}[r''] \overset{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$ to conclude $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$, for some r', M such that $U_{p,q}[r''] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r']$ and $\overline{L} \cap M = \emptyset$. The required result $U_{p,q}[r] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r']$ follows immediately.
(ii) $\mathcal{K}'' \equiv (\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r]$.
As $(\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \overset{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$, it is easy to see that there exist processes $\mathcal{K}_1$ and $\mathcal{K}_2$ such that $(\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \overset{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}_1 \xrightarrow{d_L} \mathcal{K}_2 \overset{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$. Considering transition $\mathcal{K}_1 \xrightarrow{d_L} \mathcal{K}_2$, we conclude that $\mathcal{K}_1$ has to be of the form $(\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$ and that $\mathcal{K}_2$ has to be of the form $\mathcal{Q}_{p,q} \mid U_{p,q}[r_1]$, for some process r_1. Now we have the following sequence of transitions:

$$\begin{aligned} \mathcal{Q}_{p,q} \mid U_{p,q}[r] &\xrightarrow{\tau} (\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r] \\ &\overset{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1] \\ &\xrightarrow{d_L} \mathcal{Q}_{p,q} \mid U_{p,q}[r_1] \overset{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}' . \end{aligned}$$

Applying Lemma 28(7) to transition $\underline{d}_L.\mathcal{Q}_{p,q} \mid U_{p,q}[r] \overset{\epsilon}{\Rightarrow}_{\mathrm{n}} (\langle L \rangle + \underline{d}_L.\mathcal{Q}_{p,q}) \mid U_{p,q}[r_1]$, we obtain $U_{p,q}[r] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r']$ for some M such that $\overline{L} \cap M = \emptyset$. Further, by applying Lemma 28(3) to transition $\mathcal{Q}_{p,q} \mid U_{p,q}[r_1] \overset{\epsilon}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'$, we get $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r_1 \overset{\epsilon}{\Rightarrow} r'$. Then it is obvious that $U_{p,q}[r_1] \overset{\epsilon}{\Rightarrow} U_{p,q}[r']$ and $U_{p,q}[r_1] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r']$, by using Lemma 12(1). The required result $U_{p,q}[r] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r']$ follows immediately.

(b) $\mathcal{K}_{p,q}[r] \overset{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}'' \xrightarrow{\tau} \mathcal{K}'$.
Since $\mathcal{K}' \overset{c}{\Rightarrow}_{\mathrm{n}}$ and $\mathcal{K}'' \xrightarrow{\tau} \mathcal{K}'$, it is immediate that $\mathcal{K}'' \overset{c}{\Rightarrow}_{\mathrm{n}}$ as well. Now we apply the induction hypothesis to transition $\mathcal{K}_{p,q}[r] \overset{d_L}{\Rightarrow}_{\mathrm{n}} \mathcal{K}''$ to conclude that $\mathcal{K}'' \equiv \mathcal{K}_{p,q}[r'']$, for some r'', M such that $U_{p,q}[r] \overset{\epsilon}{\underset{M}{\Longrightarrow}} U_{p,q}[r'']$ and $\overline{L} \cap M = \emptyset$. Hence, $\mathcal{K}_{p,q}[r''] \xrightarrow{\tau} \mathcal{K}'$. We apply Lemma 28(3) to this transition to get $\mathcal{K}' \equiv \mathcal{K}_{p,q}[r']$ for some r' such that $r_1 \overset{\epsilon}{\Rightarrow} r'$. The rest of the proof proceeds as in Case (8).

This concludes the proofs of the induction step and of Lemma 28(8). □

Appendix C. Proof: axiomatizing prioritized observation equivalence

In this section, we prove the second statement of Lemma 32. Let p and q be finite processes such that $I(p) \subseteq I(q)$ and such that either both are patient or neither is patient. W.l.o.g. we assume that p and q are normal forms $\sum_{i=1}^{m} \alpha_i.p_i$ and $\sum_{j=1}^{n} \beta_j.q_j$, respectively. The proof is presented by induction upon the cardinality of $I(p)$.

As the base case, suppose $I(p)$ is empty. Then p can be either *nil*, or $\sum_{i=1}^{m} \tau.p_i$ with $m > 0$, or $\sum_{i=1}^{m} \underline{\tau}.p_i$ with $m > 0$.

(1) $p \equiv nil$. The proof for this case splits into two cases depending upon whether $n = 0$:
Suppose $n = 0$, i.e., $q \equiv nil$, and take $p' \equiv q' \equiv nil$. It is obvious that $p \simeq p'$ and $q \simeq q'$. By the reflexivity law of inequational reasoning we have $\vdash p' \sqsubseteq_i q'$.
Suppose $n > 0$, and take $p' \equiv \sum_{i=1}^{n} nil$ and $q' \equiv q$. Again, it is obvious that $p \simeq p'$. Since p is patient, by given conditions, q is patient, too, and hence none of the $\beta_j.q_j$ is of the form $\underline{\tau}.q_j$. Consequently, using

Axiom iA3, $\vdash nil \sqsubseteq_i \beta_j.q_j$, for $1 \leq j \leq n$. Summing over all j and using the substitutivity law of inequational reasoning we obtain $\vdash \sum_{i=1}^{n} nil \sqsubseteq_i \sum_{j=1}^{n} \beta_j.q_j$.

(2) $p \equiv \sum_{i=1}^{m} \tau.p_i$ with $m > 0$.
Take $p' \equiv \sum_{i=1}^{m} \tau.p_i + \sum_{i=1}^{n} nil$ and $q' \equiv (\sum_{i=1}^{m} nil) + \sum_{j=1}^{n} \beta_j.q_j$. It is obvious that $p \simeq p'$ and $q \simeq q'$. If $p' \equiv \sum_{k=1}^{m+n} p'_k$ and $q' \equiv \sum_{k=1}^{m+n} q'_k$, then we can prove $\vdash p'_k \sqsubseteq_i q'_k$ using Axiom iA2, for $1 \leq k \leq m$. Since p is patient, by given conditions, q is also patient and none of the $\beta_j.q_j$ is of the form $\underline{\tau}.q_j$. Thus, when $m+1 \leq k \leq m+n$, we can prove $\vdash p_k \sqsubseteq_i q_k$ using Axiom iA3. Summing over all k and using the substitutivity law of inequational reasoning, we obtain $\vdash p' \sqsubseteq_i q'$.

(3) $p \equiv \sum_{i=1}^{m} \underline{\tau}.p_i$ with $m > 0$.
Since p is not patient, neither is q. Hence, $n > 0$ and at least one of the β_j's is $\underline{\tau}$. Let $l =_{\mathrm{df}} |\{\, j \mid \beta_j = \underline{\tau} \,\}|$ be the number of summands of q that are prefixed by $\underline{\tau}$. Since q is impatient, $l > 0$. Let further $1 \leq k \leq n$ be such that $\beta_k \equiv \underline{\tau}$. We define processes p^1 and q^1 that are strongly bisimilar to p and q, respectively, such that they have the same number of overall summands and also the same number of summands that are prefixed by $\underline{\tau}$. This is accomplished by adding some *nil* terms and by "cloning" some specific summands prefixed by $\underline{\tau}$ as follows:

- If $l = m$, then take $p^1 =_{\mathrm{df}} p\{+\sum_{i=1}^{n-l} nil\}$ and $q^1 =_{\mathrm{df}} q$.
- If $l > m$, then take $p^1 =_{\mathrm{df}} p + (\sum_{i=1}^{l-m} \underline{\tau}.p_1)\{+(\sum_{i=1}^{n-l} nil)\}$ and $q^1 =_{\mathrm{df}} q$.
- If $l < m$, then take $p^1 =_{\mathrm{df}} p\{+(\sum_{i=1}^{n-l} nil)\}$ and $q^1 =_{\mathrm{df}} q + (\sum_{i=1}^{m-l} \underline{\tau}.q_k)$.

Here we use the notation "$p\{+q\}$" to denote the term p, if q is *nil*, and $p+q$, otherwise. Let p' and q' be the processes obtained from p^1 and q^1, respectively, by reordering the summands so that all the summands prefixed by $\underline{\tau}$ precede those that do not. It is easy to see that $p \simeq p'$ and $q \simeq q'$. Now, using arguments similar to those presented in the previous cases, we can prove $\vdash p' \sqsubseteq_i q'$.

This concludes the proof of the base case with $I(p)$ empty.

For the induction step we know $|I(p)| > 0$ and thus can choose $\lambda \in I(p)$. Since $I(p) \subseteq I(q)$, we further have $\lambda \in I(q)$. Let $L_p =_{\mathrm{df}} \{\, i \mid \alpha_i \equiv \lambda \,\}$ and $L_q =_{\mathrm{df}} \{\, j \mid \beta_j \equiv \lambda \,\}$, and define $l_p =_{\mathrm{df}} |L_p|$ and $l_q =_{\mathrm{df}} |L_q|$. Further, let p^1 and q^1 be the processes obtained by reordering the summands of the processes p and q, respectively, such that the summands prefixed by λ precede those that do not. Formally, let $p^1 =_{\mathrm{df}} p_1^1 + p_2^1$, where $p_1^1 =_{\mathrm{df}} \sum_{i \in L_p} \alpha_i.p_i$ and $p_2^1 =_{\mathrm{df}} \sum_{i \notin L_p} \alpha_i.p_i$, and $q^1 \equiv q_1^1 + q_2^1$, where $q_1^1 =_{\mathrm{df}} \sum_{j \in L_q} \beta_j.q_j$ and $q_2^1 =_{\mathrm{df}} \sum_{j \notin L_q} \beta_j.q_j$. We may apply the induction hypothesis to p_2^1 and q_2^1 to obtain processes p'_2 and q'_2 such that $p_2^1 \simeq p'_2$, $q_2^1 \simeq q'_2$ and $\vdash p'_2 \sqsubseteq_i q'_2$. Next, we define processes p'_1 and q'_1 such that they have an equal number of summands and are prioritized strong bisimilar to p_1^1 and q_1^1, respectively:

- If $l_p = l_q$, then take $p'_1 =_{\mathrm{df}} p_1^1$ and $q'_1 =_{\mathrm{df}} q_1^1$.
- If $l_p > l_q$, then take $p'_1 =_{\mathrm{df}} p_1^1$ and $q'_1 =_{\mathrm{df}} q_1^1 + (\sum_{j=1}^{l_p - l_q} \beta_k.q_k)$, where $k \in L_q$.
- If $l_p < l_q$, then take $q'_1 =_{\mathrm{df}} q_1^1$ and $p'_1 =_{\mathrm{df}} p_1^1 + (\sum_{i=1}^{l_q - l_p} \alpha_k.p_k)$, where $k \in L_p$.

It is easy to show that $\vdash p'_1 \sqsubseteq_i q'_1$. Thus, if we take $p' =_{\mathrm{df}} p'_1 + p'_2$ and $q' =_{\mathrm{df}} q'_1 + q'_2$, then $p \simeq p'$, $q \simeq q'$, and $\vdash p' \sqsubseteq_i q'$. □

Appendix D. Computing prioritized observation equivalence

In this appendix we provide an alternative characterization of prioritized weak bisimulation, which has a couple of benefits. Firstly, the characterization shows how traditional *partition-refinement algorithms* for computing bisimulation [22,28] can be used to compute our prioritized observation equivalence $\approxeq$. We have implemented this approach in the *NC Concurrency Workbench* [14]. Secondly, the characterization immediately yields a *logical* characterization of $\approxeq$ in terms of *Hennessy–Milner logic* [18,25], and thus bridges the gap to

temporal logics. We start off with a definition for weak ϵ-transitions that is not parameterized by initial action sets.

Definition 41. We write $p \stackrel{\epsilon}{\Rightarrow} p'$ for processes p, p', if $p \stackrel{\epsilon}{\Rightarrow} q$ and $q \stackrel{\epsilon}{\Longrightarrow}_{I(q)} p'$ for some patient process q.

This definition permits the following characterization of prioritized weak bisimulation relations.

Proposition 42. *A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a prioritized weak bisimulation relation if and only if, for all $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, the following holds*:

(1) $p\downarrow\downarrow$ *implies* $q\downarrow\downarrow$.
(2) $p \stackrel{\alpha}{\rightarrow} p'$ *implies* $\exists q'.\, q \stackrel{\hat{\alpha}}{\Rightarrow} q'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.
(3) $q \stackrel{\alpha}{\rightarrow} q'$ *implies* $\exists p'.\, p \stackrel{\hat{\alpha}}{\Rightarrow} p'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.

Proof. For the "if" part of the proposition, let $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ be a relation that satisfies the three stated conditions. To show that $\mathcal{R}$ is a prioritized weak bisimulation relation, it suffices to prove that $p \stackrel{\tau}{\rightarrow} p'$ implies the existence of some q' such that $q \stackrel{\epsilon}{\Longrightarrow}_{I(p)} q'$ and $\langle p', q'\rangle \in \mathcal{R}$. Let $p \stackrel{\tau}{\rightarrow} p'$. By the second condition and by Def. 41 we know of a patient q_1 such that $q \stackrel{\epsilon}{\Rightarrow} q_1 \stackrel{\epsilon}{\Longrightarrow}_{I(q_1)} p'$ and $\langle p', q'\rangle \in \mathcal{R}$. Using arguments similar to those used in the proof of Lemma 15, it is easy to see that $\langle p, q_1\rangle \in \mathcal{R}$ and $I(p) = I(q_1)$. Consequently, $q \stackrel{\epsilon}{\Rightarrow} q_1 \stackrel{\epsilon}{\Longrightarrow}_{I(p)} q'$, i.e., $q \stackrel{\epsilon}{\Longrightarrow}_{I(p)} q'$, and $\langle p', q'\rangle \in \mathcal{R}$, as desired. The proof of the "only if" part is quite similar and thus omitted here. □

Given this proposition, the proof of the following characterization theorem is straightforward and analogue to similar proofs conducted in [25].

Theorem 43 (Characterization of $\approx$)**.**
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a prioritized weak bisimulation relation if and only if, for all $\langle p, q\rangle \in \mathcal{R}$ and $\alpha \in (\mathcal{A} \cup \{\epsilon, \underline{\epsilon}\}) \setminus \{\tau, \underline{\tau}\}$, the following holds:

(1) $p\downarrow\downarrow$ *implies* $q\downarrow\downarrow$.
(2) $p \stackrel{\alpha}{\Rightarrow} p'$ *implies* $\exists q'.\, q \stackrel{\alpha}{\Rightarrow} q'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.
(3) $q \stackrel{\alpha}{\Rightarrow} q'$ *implies* $\exists p'.\, p \stackrel{\alpha}{\Rightarrow} p'$ *and* $\langle p', q'\rangle \in \mathcal{R}$.

Observe that the characterization theorem uses the same transition relation on the left-hand sides and right-hand sides of the second and third conditions. It thus lends itself immediately to applying the partition-refinement algorithms of [22,28] and the logic-characterization approach of [18,25].

References

[1] L. Aceto, T. Chen, W. Fokkink, A. Ingólfsdóttir, On the axiomatizability of priority, in: Automata, Languages and Programming (ICALP 2006), LNCS, vol. 4052, Springer-Verlag, 2006.
[2] L. Aceto, W. Fokkink, A. Ingólfsdóttir, S. Nain, Bisimilarity is not finitely based over BPA with interrupt, in: Algebra and Coalgebra in Computer Science (CALCO 2005), LNCS, vol. 3629, Springer-Verlag, 2005.
[3] J. Baeten, J. Bergstra, J. Klop, Syntax and defining equations for an interrupt mechanism in process algebra, Fundamenta Informaticae vol. IX (1986) 127–168.
[4] J. Bergstra, C. Middelburg, Preferential choice and coordination conditions, Journal of Logic and Algebraic Programming 70 (2) (2007) 172–200.
[5] in: J. Bergstra, A. Ponse, S. Smolka (Eds.), Handbook of Process Algebra, Elsevier Science, 2001.
[6] G. Bhat, R. Cleaveland, G. Lüttgen, A practical approach to implementing real-time semantics, Annals of Software Engineering 7 (1999) 127–155.

[7] R. Bol, J. Groote, The meaning of negative premises in transition system specification, in: Automata, Languages and Programming (ICALP '91), LNCS, vol. 510, Springer-Verlag, 1991.
[8] M. Bravetti, R. Gorrieri, A complete axiomatization for observational congruence of prioritized finite-state behaviors, in: Automata, Languages and Programming (ICALP 2000), LNCS, vol. 1853, Springer-Verlag, 2000.
[9] J. Camilleri, G. Winskel, CCS with priority choice, Information and Computation 116 (1) (1995) 26–37.
[10] R. Cleaveland, M. Hennessy, Priorities in process algebras, Information and Computation 87 (1/2) (1990) 58–77.
[11] R. Cleaveland, G. Lüttgen, M. Mendler, An algebraic theory of multiple clocks, in: Concurrency Theory (CONCUR '97), LNCS, vol. 1243, Springer-Verlag, 1997.
[12] R. Cleaveland, G. Lüttgen, V. Natarajan, A process algebra with distributed priorities, Theoretical Computer Science 195 (2) (1998) 227–258.
[13] R. Cleaveland, G. Lüttgen, V. Natarajan, Priority in process algebra, in: Bergstra et al. [[5]], pp. 711–765.
[14] R. Cleaveland, V. Natarajan, S. Sims, G. Lüttgen, Modeling and verifying distributed systems using priorities: a case study, Software-Concepts and Tools 17 (2) (1996) 50–62.
[15] R. De Nicola, M. Hennessy, Testing equivalences for processes, Theoretical Computer Science 34 (1983) 83–133.
[16] R. Gerber, I. Lee, A resourced-based prioritized bisimulation for real-time systems, Information and Computation 113 (1) (1994) 102–142.
[17] R. Glabbeek, W. Weijland, Branching time and abstraction in bisimulation semantics, in: Information Processing '89, Elsevier Science, 1989.
[18] M. Hennessy, R. Milner, Algebraic laws for nondeterminism and concurrency, Journal of ACM 32 (1) (1985) 137–161.
[19] H. Hermanns, M. Lohrey, Priority and maximal progress are completely axiomatisable, in: Concurrency Theory (CONCUR '98), LNCS, vol. 1466, Springer-Verlag, 1998.
[20] A. Jeffrey, Translating timed process algebra into prioritized process algebra, in: Real-Time and Fault-Tolerant Systems (FTRTFT '92), LNCS, vol. 571, Springer-Verlag, 1992.
[21] C.-T. Jensen, Prioritized and independent actions in distributed computer systems, Ph.D. thesis, Aarhus University, Denmark (1994).
[22] P. Kanellakis, S. Smolka, CCS expressions, finite state processes, and three problems of equivalence, Information and Computation 86 (1) (1990) 43–68.
[23] G. Lüttgen, Pre-emptive modeling of concurrent and distributed systems, Ph.D. thesis, University of Passau, Germany, Shaker Verlag (1998).
[24] G. Lüttgen, W. Vogler, Bisimulation on speed: worst-case efficiency, Information and Computation 191 (2) (2004) 105–144.
[25] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[26] R. Milner, A complete axiomatisation for observational congruence of finite-state behaviours, Information and Computation 81 (2) (1989) 227–247.
[27] V. Natarajan, L. Christoff, I. Christoff, R. Cleaveland, Priorities and abstraction in process algebra, in: Foundations of Software Technology and Theoretical Computer Science (FSTTCS '94), LNCS, vol. 880, Springer-Verlag, 1994.
[28] R. Paige, R. Tarjan, Three partition refinement algorithms, SIAM Journal of Computing 16 (6) (1987) 973–989.
[29] I. Phillips, CCS with priority guards, in: Concurrency Theory (CONCUR 2001), LNCS, vol. 2154, Springer-Verlag, 2001, a revised version of this paper has been accepted for publication in the Journal of Logic and Algebraic Programming.
[30] K. Prasad, Broadcasting with priority, in: Europ. Symp. Programming (ESOP '94), LNCS, vol. 788, Springer-Verlag, 1994.
[31] D. Sangiorgi, R. Milner, The problem of 'weak bisimulation up to', in: Concurrency Theory (CONCUR '92), LNCS, vol. 630, Springer-Verlag, 1992.
[32] C. Verhoef, A congruence theorem for structured operational semantics with predicates and negative premises, Nordic Journal of Computing 2 (2) (1995) 274–302.