January 1990

A Resource-Based Prioritized Bisimulation for Real-Time Systems

Richard Gerber
University of Pennsylvania

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Abstract
The behavior of concurrent, real-time systems can be specified using a process algebra called CCSR. The underlying computation model of CCSR is resource-based, in which multiple resources execute synchronously, while processes assigned to the same resource are interleaved according to their priorities. CCSR allows the algebraic specification of timeouts, interrupts, periodic behaviors and exceptions. This paper develops a natural treatment of preemption, which is based not only on priority, but also on resource utilization and inter-resource synchronization. The preemption ordering leads to a term equivalence based on strong bisimulation, which is also a congruence with respect to the operators. Consequently the equivalence yields a compositional proof system, which is illustrated in the verification of resource-sharing, producer-consumer problem.

A Resource -Based Prioritized Bisimulation For Real-Time Systems

MS-CIS-90-69
GRASP LAB 235

Richard Gerber
Insup Lee

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Revised
March 1992

A Resource-Based Prioritized Bisimulation for Real-Time Systems*

Richard Gerber
Dept. of Computer Science
University of Maryland
College Park, MD 20742
rich@cs.umd.edu

Insup Lee
Dept. of Computer and Info. Science
University of Pennsylvania
Philadelphia, PA 19104
lee@cis.upenn.edu

January 30, 1992

*This research was supported in part by ONR N00014-89-J-1131 and DARPA/NSF CCR90-14621.

Abstract

The behavior of concurrent, real-time systems can be specified using a process algebra called CCSR. The underlying computation model of CCSR is resource-based, in which multiple resources execute synchronously, while processes assigned to the same resource are interleaved according to their priorities. CCSR allows the algebraic specification of timeouts, interrupts, periodic behaviors and exceptions. This paper develops a natural treatment of preemption, which is based not only on priority, but also on resource utilization and inter-resource synchronization. The preemption ordering leads to a term equivalence based on strong bisimulation, which is also a congruence with respect to the operators. Consequently the equivalence yields a compositional proof system, which is illustrated in the verification of resource-sharing, producer-consumer problem.

1 Introduction

The timing behavior of a real-time system depends not only on delays due to process synchronization, but also on the availability of shared resources. Most current real-time models adequately capture delays due to process synchronization; however, they abstract out resource-specific details by assuming idealistic operating environments. On the other hand, scheduling and resource allocation algorithms used for real-time systems ignore the effect of process synchronization except for simple precedence relations between processes. What is needed is a theory that combines the areas of formal specification and real-time scheduling, and thus, can help us reason about systems that are sensitive to deadlines, process interaction and resource availability.

Our approach to this problem is a process algebra called the *Calculus for Communicating Shared Resources*, or CCSR. The CCSR computation model reflects a resource-based philosophy regarding real-time concurrency. Within this approach, a real-time system is composed of one or more resources, each of which is inherently sequential in nature. Thus, while many processes may share a single resource, at any point in time, a resource only has the capacity to execute a solitary event from one of the processes. This constraint quite naturally leads to an interleaving notion of concurrency at the resource level of the system. A priority ordering is used to arbitrate between simultaneous resource requests. At the system level, lock-step parallelism occurs when a group of resources are executed simultaneously.

Strongly influenced by SCCS (Milner, 1983), CCSR is a process algebra that uses a synchronous form of concurrency, and possesses a term equivalence based on a prioritized version of strong bisimulation (Park, 1981). The development of the equivalence relation mandates a treatment of preemption based not only on priority, but also on resource utilization and inter-resource synchronization.

The challenge of suitably defining preemption can be illustrated by a small example. Consider a process P, which is hosted on some resource r. Assume that during its first time unit P may either execute $a?$ and move to a state P_1, execute $b?$ and move to a state P_1, or idle and re-enable P. In the CCSR language, the process P is rendered as follows:

$$P \overset{\text{def}}{=} \{a?\} : P_1 \;+\; \{b?\} : P_2 \;+\; \emptyset : P$$

Also, assume a synchronization paradigm similar to that of CSP (Hoare, 1985); i.e., $a?$ or $b?$ may execute if and only if there is a simultaneous occurrence of $a!$ or $b!$, respectively. The actual first execution of P depends heavily on the context in which it is placed. Factors include whether the context offers either $a!$, $b!$ or both events, the priorities of $a?$, $b?$, $a!$ and $b!$, or whether P may be blocked by another process that requires its resource (in which case the idle branch would be taken). In defining an adequate notion of preemption, we must consider all of these factors.

As we show in this paper, our preemption ordering leads to several desirable properties, not the least of which being an equivalence which is also a congruence. Based on these results, we have developed a compositional proof system for CCSR, which facilitates the algebraic verification of real-time systems.

The remainder of this paper is organized as follows. In Section 2 we develop our computation model. In Section 3 we introduce the CCSR language, and provide its informal semantics. Then, in Section 4, we motivate our theoretical treatment of CCSR by presenting a real-time, resource-sharing example whose correct temporal behavior depends on priority. In Section 5 we develop CCSR's semantic theory, which we use in Section 6 to present the proof system. Then in Section 7 we return to our example, and prove its correctness with respect to our proof system. In Section 8 we compare our approach to related research in the field, and in Section 9 we conclude, and remark on the significance of this work.

2 The Computation Model

The basic unit of computation is the *event*, which is used to model both local resource execution as well as inter-resource synchronization. When executed by a resource, each event consumes exactly one time unit. We let Σ represent the universal set of events.

Since a system potentially consists of many resources, multiple events may occur at any time throughout the course of its execution. We call such occurrences *actions*, and they are represented by sets in $\mathbb{P}(\Sigma)$. As in SCCS (Milner, 1983), the passage of time is implicitly captured by a sequence of actions, where one clock "tick" corresponds to the execution of a single action. In general, we let the letters a, b and c range over the event set Σ, and the letters A, B and C range over the action set $\mathbb{P}(\Sigma)$. Also, we let the letter ϕ range over renaming functions on Σ; that is, $\phi \in \Sigma \rightarrow \Sigma$. We overload notation and extend such functions to sets in the usual way, where $\phi(A) = \{\phi(a) \mid a \in A\}$.

Termination. The termination event, or "$\surd$", has the unique property that it is not "owned" by any particular resource. In the spirit of CSP (Hoare, 1985), if $\surd \in A$ for some action A, this means that the system executing A is capable of terminating. Also, $\surd$ is a fixed point of all event renaming functions $\phi \in \Sigma \rightarrow \Sigma$; i.e., for all such ϕ, $\phi(\surd) = \surd$.

Resources and Actions. We consider individual resources to be inherently sequential in nature. That is, a single resource is capable of synchronously executing actions that consist, *at most*, of a single event. Actions that consist of multiple events must be formed by the synchronous execution of multiple resources. We denote $\mathcal{R}$ to represent the set of resources available to a system, and let i, j, and k range over $\mathcal{R}$.

This notion of execution leads to a natural partition of $\Sigma - \{\surd\}$ into mutually disjoint subsets, each of which can be considered the set of events available to a single resource. For all i in $\mathcal{R}$ we denote Σ_i as the collection of events exclusively "owned" by resource i:

$$\forall i \in \mathcal{R}, \forall j \in \mathcal{R} \,.\, i \neq j,\ \Sigma_i \cap \Sigma_j = \emptyset$$

This type of alphabet partitioning is similar to that found in the I/O Automata model (Lynch and Tuttle, 1988), where it is used to define a notion of fairness. However, here it is used to help mandate our resource-induced mutual exclusion condition. As we have stated, a single resource is capable of executing actions that consist of at most one event. Extrapolating this principle to a system of concurrent resources, the CCSR action domain is defined as follows:

$$\mathcal{D} \;=\; \{A \in p(\Sigma) \,|\, \forall i \in \mathcal{R},\ |A \cap \Sigma_i| \leq 1\}$$

That is, an action executable by a CCSR term may consist of at most one event from each component resource. Here, "$p(\Sigma)$" denotes the set of finite subsets of Σ, and "$|S|$" denotes the cardinality of a finite set "S".

For a given action A, we use the notation $\rho(A)$ to represent the set of resources that execute the events in A: $\rho(A) = \{i \in \mathcal{R} \,|\, \Sigma_i \cap A \neq \emptyset\}$. Note that since for all i, $\surd \notin \Sigma_i$, $\rho(A) = \rho(A - \{\surd\})$.

2.1 Priority

At any point in time many events may be competing for the ability to execute on a single resource. We help arbitrate such competition through the use of a priority ordering over Σ. There is a finite range of priorities at which events may execute. Letting mp be the maximum possible priority, we denote $PRI = \{0, \ldots, mp\} \subseteq \mathbb{N}$ as the set of priorities available to events in the system. Thus we can order the events in Σ by a priority mapping $\pi \in \Sigma \rightarrow PRI$.[1]

Using π, we define the preorder "$\leq_p$" that reflects the notion of priority over the domain

[1]As one the referees has pointed out, the theory of CCSR remains valid even if we assume that event-wise priorities are partially ordered.

$$\begin{tikzcd}
 & & \{b,d\} & & \\
 & \{b,c\} \arrow[ur] & & \{a,d\} \arrow[ul] & \\
\{b\} \arrow[ur] & & \{a,c\} \arrow[ul] \arrow[ur] & & \{d\} \arrow[ul] \\
 & \{a\} \arrow[ul] \arrow[ur] & & \{c\} \arrow[ul] \arrow[ur] & \\
 & & \emptyset \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Figure 1: Example of the Priority Ordering, $<_p$

$\mathcal{D}$. For all $A, B \in \mathcal{D}$, $A \leq_p B$ if and only if for all i in $\rho(A) \cup \rho(B)$,

$$\begin{array}{l}
A \cap \Sigma_i = \emptyset \lor \\
(\exists a.\ A \cap \Sigma_i = \{a\} \land \pi(a) = 0) \lor \\
(\exists a \exists b.\ A \cap \Sigma_i = \{a\} \land B \cap \Sigma_i = \{b\} \land \pi(a) \leq \pi(b))
\end{array}$$

Note that the first disjunct establishes that idling (i.e., the execution of *no* event) has the lowest priority on every resource. Also, the second disjunct accounts for the possibility that an event may have a priority of 0.

Based on this definition, we use the notation "$A <_p B$" to represent that A has lower priority than B; i.e., $A \leq_p B$ and $B \not\leq_p A$.

Example 2.1 Consider the events a, b, c and d, where

1. The resource mapping is: $\rho(\{a, b\}) = \{r1\}$ and $\rho(\{c, d\}) = \{r2\}$.

2. The event-wise priority is: $\pi(a) < \pi(b)$ and $\pi(c) < \pi(d)$;
   e.g., $\pi(a) = 1$, $\pi(b) = 2$, $\pi(c) = 2$ and $\pi(d) = 3$.

Then Figure 1 illustrates the ordering between the different *feasible* actions that can by formed among these events. In the figure, the arrow "$\longrightarrow$" is transitive, as it represents the "$<_p$" relation. Thus the "$\emptyset$" action – denoting both resources in an idle state – has the lowest priority, while the $\{b, d\}$ action has the highest. $\square$

2.2 Synchronization

In CCSR, the lowest form of communication is accomplished through the simultaneous execution of synchronizing events. The model treats such synchronizing events as being statically "bound" together by the various connections between system resources. To capture this property we make use of what we call *connection sets.* A connection set is a set of events that exhibits the "all or none" property of event synchronization: At time t, if any of the events in a given connection set wish to execute, they all must execute. A familiar example of this concept can be drawn from CSP (Hoare, 1978), where the alphabet of events is $\{c_1!, c_1?, c_2!, c_2?, c_3!, c_3?, \ldots\}$, where "$c_i$" is a channel, "$c_i!$" is interpreted as a write action, and "$c_i?$" is interpreted as a read action. When a read and a write occur simultaneously on the same channel, the communication is considered successful. The connection sets in such languages are simply $\{c_1!, c_1?\}$, $\{c_2!, c_2?\}$, etc. More formally, a connection set is an equivalence class formed by the equivalence relation "$\bullet\!\!-\!\!\bullet$".

Definition 2.1 $\bullet\!\!-\!\!\bullet \subseteq \Sigma \times \Sigma$ *is an equivalence relation, where* $a \bullet\!\!-\!\!\bullet b$ *denotes that* a *is connected to* b*. We use the notation* $connections(a)$ *to represent the equivalence class (or connection set) of* a*, and we stipulate that for all* $a \in \Sigma$*,* $connections(a) \in \mathcal{D}$*.* □

The reason for this last constraint is straightforward. If a set of events is fully connected, it should be able to execute, and therefore be in the action domain. We note that this generalized notion of synchronization is similar to that found in Arnold (1982), in which a synchronized behavior is a set of events that must be executed simultaneously.

Of course, if an event is used solely to model a resource's local computation, it need not synchronize with any other event in the system. Such events occupy their own (singleton) connection set. Also, the terminating event "$\surd$" belongs to its own connection set, as successful termination does not require the explicit synchronization of resources.

We use the notation $Connections(A)$ to assemble all of the connection sets represented in A:

$$Connections(A) = \bigcup_{a \in A} connections(a)$$

Thus, a set A is *fully synchronized* if it can be fully decomposed into a set of the connection sets (or it is empty). We use the predicate $fullsync(A)$ to represent this:

$$fullsync(A) \quad \text{iff} \quad A = Connections(A)$$

Note that if A is fully synchronized, it requires no additional communicating partners. Appealing to the example of CSP, we would say that the action $\{a!, a?\}$ is fully synchronized.

In the process of reasoning about a large system we often decompose it into smaller components, and then attempt to reason about the components. For example, assume that such a subsystem is hosted on a resource set I, and that the subsystem executes some action A (i.e., $\rho(A) \subseteq I$). When analyzing only the subsystem, we do not need to know that A is fully synchronized – indeed, more system resources may be needed to achieve this result. Instead, we wish to determine whether A is synchronized with respect to the resource set I; that is, whether $sync_{(I)}(A)$ holds, where

$$sync_{(I)}(A) \quad \text{iff} \quad A = Connections(A) \cap (\textstyle\bigcup_{i \in I} \Sigma_i \cup \{\surd\})$$

If $sync_{(I)}(A)$ holds, A cannot make any additional connections with the resources in I.

Finally, it is often convenient to decompose an action A into two parts: that which is fully synchronized (or resolved), and that which is not (or still unresolved). To do this, we make use of the following two definitions:

$$\begin{aligned} res(A) &= \bigcup\{B \subseteq A \mid fullsync(B)\} \\ unres(A) &= A - res(A) \end{aligned}$$

Again using the CSP-like notation, if $A = \{a!, a?, b!, b?, c!, d?\}$, we have $res(A) = \{a!, a?, b!, b?\}$ and $unres(A) = \{c!, d?\}$.

2.3 Priority-Canonical Events

In an unprioritized calculus such as SCCS (Milner, 1983), the idle action serves two distinctly different functions. One is to denote pure idling; for example, the SCCS term $1 : (P \times Q)$ represents a process that idles for one time unit, and subsequently executes the term $P \times Q$. On the other hand, the idle can also denote the combined actions of two communicating partners; e.g., the term $(a : P) \times (\overline{a} : Q)$ is strongly equivalent to $1 : (P \times Q)$; i.e., "pure" idling and successful communication are represented in the same manner.

In a prioritized, resource-based algebra such as CCSR, it would be difficult to justify the abstraction of either priority or resource usage. For example, assume that at time 1 the process P executes the action $\{a!, a?\}$; where $fullsync(\{a!, a?\})$, $\pi(a!) = 1$, and $\pi(a?) = 2$; with $\rho(\{a!\}) = \{r1\}$ and $\rho(\{a?\}) = \{r2\}$.

Now assume that $\{a!, a?\}$ could be abstracted from the behavior of P; that is, it could be mapped to the idle action $\emptyset$ while preserving the same resource untilization. But then another process Q, concurrently running with P, could also utilize the resources $r1$ and $r2$ at time 1. This would violate the defining principle of CCSR – that a resource may execute only one event at a time. For a similar reason we do not abstract priority information from a system's behavior.

While we are constrained by these limitations, we may still "hide" an event up to its priority and resource usage. Note that these factors naturally partition $\Sigma - \{\surd\}$ into equivalence classes; that is, the events a and b are in the same class if and only if $\rho(\{a\}) = \rho(\{b\})$ and $\pi(a) = \pi(b)$. In CCSR, we use the symbol "τ_i^n" to denote a canonical representative event from each class, where τ_i^n is mapped to resource i and has priority n.

Consider the difference between the actions "$\emptyset$" and "$\{\tau_i^0\}$." While $\emptyset =_p \{\tau_i^0\}$, an execution of $\emptyset$ denotes that resource i (and, in fact, all other resources) have been released for use by other processes. On the other hand, an occurrence of $\{\tau_i^0\}$ denotes that resource i is being used. For example, if we were to hide some action $\{a\}$, where a has a priority of 0 and is hosted on resource i, the result would be $\{\tau_i^0\}$.

To implement this type of abstraction, we introduce a unique renaming function, ϕ_π, such that if $\rho(\{a\}) = i$ and $\pi(a) = n$, then $\phi_\pi(a) = \tau_i^n$. It follows that the τ_i^n are fixed-points of priority renaming; that is, $\phi_\pi(\tau_i^n) = \tau_i^n$. All such canonical representatives are local with respect to their own resources; that is, they belong to their own connection sets.

3 The CCSR Language

The syntax of CCSR resembles, in some respects, that of SCCS. Let $\mathcal{E}$ represent the domain of terms, and let E, F, G and H range over $\mathcal{E}$. Additionally we assume an infinite set of free term variables, FV, with X ranging over FV and $free(E)$ representing the set of free variables in the term E. Let $\mathcal{P}$ represent the domain of closed terms, which we call *agents* or alternatively, *processes*, and let P, Q, R and S range over $\mathcal{P}$. The following grammar defines the terms of CCSR:

$$E \;:=\; NIL \mid A : E \mid E + E \mid E\,{}_I\|_J\,E \mid E \mathbin{\triangle_t^B} (E, E, E) \mid [E]_I \mid E \backslash A \mid fix(X.E) \mid X$$

We note that a CCSR term does not define the structure of its computation model; e.g., the resource set, the priority mapping and the connectivity relation. Rather, we consider the action domain to be defined separately, and then a CCSR term inherits its characteristics. In Gerber (1991) we describe an implementation of the language, in which a configuration schema is used to define elements such as $\mathcal{D}$, π and $\bullet\!\!-\!\!\bullet$.

While we give a semantics for our operators in subsequent sections, we briefly present some motivation for them here. The term NIL corresponds to **0** in SCCS – it can execute no action whatsoever. The Action operator, "$A : E$", has the following behavior. At the first time unit, the action A is executed, proceeded by the term E. The Choice operator represents selection – either of the terms can be chosen to execute, subject to the constraints of the environment. For example, the term $(A : E) + (B : F)$ may execute A and proceed to E, or it may execute B and proceed to F.

The Parallel operator $E_I \|_J F$ has two functions. It defines the resources that can be used by the two terms, and also forces synchronization between them. Here, $I \subseteq \mathcal{R}$ is a set of the resources allotted to E, and $J \subseteq \mathcal{R}$ is a set of the resource allotted to F. In the case where $I \cap J \neq \emptyset$, E and F may be able to share certain resources. But as we have stated, such resource-sharing must be interleaved.

The Scope construct $E \triangle_t^B (F, G, H)$ binds the term E by a temporal scope (Lee and Gehlot, 1985), and it incorporates both the features of timeouts and interrupts. We call t the *time bound* and B the *termination control*, where $t \in \mathbb{N}^+ \cup \{\infty\}$ (i.e., t is either a positive integer or infinity), and $B = \{\surd\}$ or $B = \emptyset$.

While E is executing we say that the scope is *active*. The scope can be exited in a number of ways, depending on the values of E, H, t and B. If E successfully terminates within time t by executing "$\surd$", then F is initiated. Here, if $B = \{\surd\}$, the transition from E to F will retain its ability to signal termination, while if $B = \emptyset$, the entire construct will terminate only when F does.

There are two other ways in which the scope may be exited. If E fails to terminate within t units, the "exception-handler" G is executed. Lastly, at any time throughout the execution of E, it may be interrupted by H, and the scope is then departed.

As an example of the Scope operator, consider the following specification: "Execute P for a maximum of 100 time units. If P successfully terminates within that time, then terminate the system. However, if P fails to finish within 100 time units, at time 101 start executing R. At any time during the execution of P, allow interruption by an action $\{a?\}$ which will halt P, and initiate the interrupt-handler S." This system may be realized by the following term: $P \triangle_{100}^{\{\surd\}} (NIL, R, \{a?\} : S)$.

Now consider this specification: "Execute P for a maximum of 100 time units. If P successfully terminates within that time, "cancel" the termination and proceed to Q. If P fails to finish within 100 time units, at time 101 start executing R." This specification yields the following term: $P \triangle_{100}^{\emptyset} (Q, R, NIL)$.

We note that sequential composition may be realized by using the Scope operator. To sequentially compose E and F, we may use this term: $E \triangle_{\infty}^{\emptyset} (F, NIL, NIL)$.

The Close operator, $[E]_I$, denotes that the term E occupies *exactly* the resources represented in the index I. In addition, Close produces a term that totally utilizes the resources in I; that

$$\boxed{\begin{array}{rcl}
Producer_1 & \stackrel{\text{def}}{=} & (\delta_\infty(\{p_1\} : \delta_2(\{int_1!\} : IDLE)))\ \Delta^\bullet_6\ (NIL, Producer_1, NIL) \\
Producer_2 & \stackrel{\text{def}}{=} & (\delta_\infty(\{p_2\} : \delta_4(\{int_2!\} : IDLE)))\ \Delta^\bullet_6\ (NIL, Producer_2, NIL) \\
Consumer_1 & \stackrel{\text{def}}{=} & \delta_\infty(\{int_1?\} : \delta_\infty(\{c_1\} : \delta_\infty(\{c_1\} : Consumer_1))) \\
Consumer_2 & \stackrel{\text{def}}{=} & \delta_\infty(\{int_2?\} : \delta_\infty(\{c_2\} : \delta_\infty(\{c_2\} : Consumer_2))) \\
System & \stackrel{\text{def}}{=} & [\ (Producer_1\,{}_{\{1\}}\|_{\{2\}}\,Producer_2)\,{}_{\{1,2\}}\|_{\{3\}} \\
 & & \quad (Consumer_1\,{}_{\{3\}}\|_{\{3\}}\,Consumer_2)\ ]_{\{1,2,3\}}
\end{array}}$$

Figure 2: Producer-Consumer System

is, it prohibits further sharing of those resources. When we present the semantics of CCSR, we shall show exactly what this means.

The Hiding operator $E \backslash A$ masks actions in E up to their resource usage and priority, in that while the actions themselves are hidden, their priorities are still observable. The term $fix(X.E)$ denotes recursion, allowing the specification of infinite behaviors.

4 An Example

We present a time-critical, Producer/Consumer example that illustrates the interrelationship between resource-sharing and priority in CCSR. This example illustrates that in some real-time applications, a system's correctness can hinge on the ability to implement priority.

The system is composed of four agents: $Producer_1$, $Producer_2$, $Consumer_1$ and $Consumer_2$. Also, there are three resources, which we call "resource 1", "resource 2" and "resource 3"; $Producer_1$ is hosted on resource 1, $Producer_2$ is hosted on resource 2, while $Consumer_1$ and $Consumer_2$ share resource 3.

We use the following notation that facilitates a concise specification of our system.

$$IDLE \stackrel{\text{def}}{=} \emptyset : IDLE$$

$$\delta_t(P) \stackrel{\text{def}}{=} \begin{cases} IDLE & \text{if } t = 0 \\ P\ +\ \emptyset : \delta_{t-1}(P) & \text{otherwise} \end{cases}$$

The $IDLE$ process executes indefinitely, without contributing any observable behavior. In $\delta_t(P)$, the initial action of P must execute within time t; otherwise the process goes into an

- Resources:
  - Resource 1: $p_1, int_1! \in \Sigma_1$
  - Resource 2: $p_2, int_2! \in \Sigma_2$
  - Resource 3: $int_1?, int_2?, c_1, c_2 \in \Sigma_3$
- Connection sets:
  - Inter-resource: $\{int_1!, int_1?\}$, $\{int_2!, int_2?\}$
  - Local: $\{p_1\}$, $\{p_2\}$, $\{c_1\}$, $\{c_2\}$
- Priorities:
  - Resource 1:$\pi(p_1) = 1,\ \pi(int_1!) = 0$
  - Resource 2:$\pi(p_2) = 1,\ \pi(int_2!) = 0$
  - Resource 3:$\pi(c_1) = 3,\ \pi(c_2) = 3, \pi(int_1?) = 2,\ \pi(int_2?) = 1$

Figure 3: System Resources, Connections and Priorities

idle state. The timeout value t may range over $\mathbb{N} \cup \{\infty\}$, i.e., if $t = \infty$, the execution of P may delay indefinitely.

The producer-consumer system is shown in Figure 2. $Producer_1$ describes a periodic process, with a period of 6 time units. Within each period, the process decides when (or if) it should enter its "production" phase. That is, it "produces" for 1 time unit by executing the action $\{p_1\}$. Then it attempts to interrupt $Consumer_1$ by executing the action $\{int_1!\}$. However, if the interrupt is not accepted within 2 time units, $Producer_1$ goes into an idle state. If the interrupt is successful, the process idles for the remainder of the period. $Producer_2$ resembles $Producer_1$ except for one fact: its interrupt must be accepted within 4 time units.

$Consumer_1$ waits indefinitely for $Producer_1$ to issue its interrupt. Then it goes into a "digestion phase" for 2 time units, during which the action $\{c_1\}$ is executed. After the second execution of $\{c_1\}$, $Consumer_1$ is restarted. $Consumer_2$ is the dual of $Consumer_1$, in that $int_2?$ signals its consumption phase, and c_2 marks its digestion phase.

The configuration of the system is shown in Figure 3. Note that $\pi(int_1!) = 0$ and $\pi(int_2!) = 0$; this means that the Consumers are solely responsible for "driving" the interrupts. Further, since $\pi(int_1?) = 2$ and $\pi(int_2?) = 1$, resource 3 "prefers" $int_1?$ over $int_2?$.

We informally state our correctness criterion as follows: During every six unit period, both producers must produce, and both consumers must consume. In Section 7 we formalize this property, and we use our set of laws to show that the system satisfies it.

5 An Operational Semantics

In this section we present an operational semantics for closed terms, in the style from Plotkin (1981). We do this in two steps. First, we define a labeled transition system $\langle \mathcal{E}, \rightarrow, \mathcal{D} \rangle$, which is a relation $\rightarrow \subseteq \mathcal{E} \times \mathcal{D} \times \mathcal{E}$. We denote each member (E, A, F) of "$\rightarrow$" as "$E \xrightarrow{A} F$". We call this transition system *unconstrained*, in that no priority arbitration is made between actions. Thus, if $E \xrightarrow{A} F$ is in "$\rightarrow$", it means that in a system without preemption constraints, a term E may execute A and proceed to F. After presenting "$\rightarrow$", we use it to define a prioritized transition system $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$, which is sensitive to preemption. This two-phased approach greatly simplifies the definition of "$\rightarrow_\pi$"; similar tactics have been used in the treatment of CCS priority (Cleaveland and Hennessy, 1990), and the definition of a maximum parallel semantics for Occam (Huizing, Gerth, and de Roever, 1987).

Throughout, we use the following notation. For a given set of resources $I \subseteq \mathcal{R}$, we let Σ_I represent the set $\bigcup_{i \in I} \Sigma_i$. Also, $A * B = (A - \{\surd\}) \cup (B - \{\surd\}) \cup (A \cap B)$; that is, the termination event "$\surd$" is an element of $A * B$ if and only if it is in both A and B.

5.1 Unconstrained Transition System

Figure 4 presents the unconstrained transition system, "$\rightarrow$". The rules for Action, Choice and Recursion are quite straightforward, as described in Section 3. The other rules, however, require some additional explanation.

Parallel. The four side conditions define both the resource mapping and synchronization constraints imposed on terms that operate in a concurrent fashion. The first two conditions define the resources on which the terms E_1 and E_2 may execute. That is, A_1 must be hosted on the resources denoted by I, while A_2 must be hosted on the resources denoted by J. Moreover, the third condition stipulates that single resources may not execute more than one event at a time.

The final condition defines our notion of inter-resource synchronization; that is, A_1 and A_2 may execute simultaneously if and only if they are connected in the following sense: If some event $a \in A_1$ is connected to an event $b \in \Sigma_J$, then b *must* appear in A_2, and *vice versa*. This synchronization constraint is a generalized version of that found in CSP.

Scope. There are four rules for the Scope operator, corresponding to the four actions that may be taken while a term E is bound by a temporal scope. Assume that $E \xrightarrow{A} E'$ with $\surd \notin A$, and that $t > 1$. In such a situation, the ScopeC law is used to keep the temporal scope active; i.e., E' is bound by the scope with its time limit decremented to $t - 1$. On the other hand if $\surd \in A$ and $t \geq 1$, ScopeE is used. In this case the scope is departed by executing

Action : $A : E \xrightarrow{A} E$

ChoiceL : $$\frac{E \xrightarrow{A} E'}{E + F \xrightarrow{A} E'}$$

ChoiceR : $$\frac{F \xrightarrow{A} F'}{E + F \xrightarrow{A} F'}$$

Parallel :
$$\frac{E_1 \xrightarrow{A_1} E_1',\ E_2 \xrightarrow{A_2} E_2'}{E_1\ {}_I\|_J\ E_2 \xrightarrow{A_1 * A_2} E_1'\ {}_I\|_J\ E_2'} \quad \left(\begin{array}{l} \rho(A_1) \subseteq I,\ \rho(A_2) \subseteq J, \\ \rho(A_1) \cap \rho(A_2) = \emptyset,\ sync_{(I \cup J)}(A_1 * A_2) \end{array} \right)$$

ScopeC : $$\frac{E \xrightarrow{A} E'}{E \,\Delta_t^B\, (F,G,H) \xrightarrow{A} E' \,\Delta_{t-1}^B\, (F,G,H)} \quad (t > 1,\ \surd \notin A)$$

ScopeE : $$\frac{E \xrightarrow{A} E'}{E \,\Delta_t^B\, (F,G,H) \xrightarrow{A * B} F} \quad (t \geq 1,\ \surd \in A)$$

ScopeT : $$\frac{E \xrightarrow{A} E'}{E \,\Delta_t^B\, (F,G,H) \xrightarrow{A} G} \quad (t = 1,\ \surd \notin A)$$

ScopeI : $$\frac{H \xrightarrow{A} H'}{E \,\Delta_t^B\, (F,G,H) \xrightarrow{A} H'} \quad (t \geq 1)$$

Close : $$\frac{E \xrightarrow{A} E'}{[E]_I \xrightarrow{A \cup (T_I^0 - T_{\rho(A)}^0)} [E']_I} \quad (\rho(A) \subseteq I)$$

Hiding : $$\frac{E \xrightarrow{B} E'}{E \backslash A \xrightarrow{\phi_{hide(A)}(B)} E' \backslash A} \quad (fullsync(A),\ fullsync(A \cap B))$$

Recursion : $$\frac{E[fix(X.E)/X] \xrightarrow{A} E'}{fix(X.E) \xrightarrow{A} E'}$$

Figure 4: Unconstrained Transition System

$A * B$, at which time F is initiated. By the definition of "$*$", if $B = \emptyset$, then $A * B = A - \{\surd\}$. That is, while E itself may terminate by executing A, the entire term will terminate when (or if) F does. But if $B = \{\surd\}$, then $A * B = A$. This means that the entire term may terminate by executing A (recall the discussion in Section 3).

Now assume that $E \xrightarrow{A} E'$ such that $\surd \notin A$, and that $t = 1$. Thus implies that the ScopeT rule must be used ("T" is for timeout). Here, the scope has "timed out", and thus, first A is executed, followed by the exception-handler G. Finally, the ScopeI rule shows that the term H may interrupt at any time while the temporal scope is active.

Close. The Close operator assigns terms to occupy *exactly* the resource set denoted by the index I. First, the action A may not utilize *more* than the resources in I; otherwise it is not admitted by the transition system. If the events in A utilize less than the set I, the action is augmented with the canonical, 0-priority events from each of the unused resources (see Section 2.3). For example, assume E executes an action A, and that there is some $i \in I$ such that $i \notin \rho(A)$. In $[E]_I$, this gap is filled by including τ_0^i in A. Here we use the notation $\mathcal{T}_J^0$ to represent *all* of the 0-priority events from the resource set J: $\mathcal{T}_J^0 = \{\tau_j^0 \mid j \in J\}$.

Close serves two important, and interrelated functions in the specification of CCSR processes. When a term E is embedded in a closed context such as $[E]_I$, we ensure that there is no further sharing of the resources in I; that is, it excludes additional interleaving concurrency on these resources. As we show in Section 5.3, this results in an increased amount of priority arbitration, i.e., a small number of possible behaviors.

Hiding. Assume that E executes an action B, and that $fullsync(A)$ and $fullsync(A \cap B)$ both hold. Then using the Hiding rule, $E \backslash A$ executes an action that reduces the events in $A \cap B$ to their "canonical" priority representation, as described in Section 2.3. If $A \subseteq \Sigma$, we construct the function $\phi_{hide(A)}$ as follows. For all a in Σ,

$$\phi_{hide(A)}(a) = \begin{cases} \phi_\pi(a) & \text{if } a \in A \\ a & \text{otherwise} \end{cases}$$

Thus, $\phi_{hide(A)}(B) = (B - A) \cup \phi_\pi(B \cap A)$; i.e., all of the events in $B \cap A$ are mapped to their corresponding "canonical" events.

The reason for this nonstandard hiding construction should be clear when viewed from the perspective of resource usage. As an example, let $a, b \in \Sigma_i$, with $\{a\}$ and $\{b\}$ as connection sets; i.e., both events are completely local to resource i. Now let $E = \{a\} : NIL$ and $F = \{b\} : NIL$. In a more "standard" definition of hiding, $E \backslash \{a\} \xrightarrow{\emptyset} NIL \backslash \{a\}$. In other words, all "$a$" is completely abstracted from the system behavior. But in this definition, we find that $(E \backslash \{a\})_{\{i\}} \|_{\{i\}} F \xrightarrow{\{b\}} (NIL \backslash \{a\})_{\{i\}} \|_{\{i\}} NIL$. This would violate the resource-based execution model, in that two events from resource i, b and a, would execute simultaneously.

Proposition 5.1 *All terms in $\mathcal{E}$ are well-defined, in that if $E \in \mathcal{E}$ and $E \xrightarrow{A} E'$, then $A \in \mathcal{D}$.*

The proof follows directly from the definition of the operators. $\square$

5.2 Preemption

The prioritized transition system is based on the notion of *preemption*, which unifies CCSR's treatment of synchronization, resource-sharing, and priority. The definition of preemption is straightforward. Let "$\prec$", called the *preemption order*, be a transitive, irreflexive, binary relation on actions. Then for two actions A and B, if $A \prec B$, we can say that "A is preempted by B". This means that in all real-time contexts, if a system can choose between executing either A or B, it will execute B. In the terminology of our calculus this can be stated as follows: The term $(A : E) + (B : F)$ can be replaced by the term $B : F$ if and only if $A \prec B$.

Since such a replacement must be valid for all possible contexts, the relation "$\prec$" must be chosen rather judiciously. Of course, one such relation is a trivial one; that is, where $A \prec B$ *never* holds for *any* actions A and B. With this definition, no preemption would ever occur, and would lead to an unprioritized calculus. Instead we wish to utilize our priority structure as much as possible and to do this we must make use of both local resource priority decisions, as well as synchronization constraints imposed by the environment.

Definition 5.1 *For all $A \in \mathcal{D}$, $B \in \mathcal{D}$, $A \preceq B$ if and only if*

$$\rho(A) = \rho(B) \ \wedge\ unres(A) = unres(B) \ \wedge\ res(A) \leq_p res(B)$$

The relation "$\preceq$" defines a preorder over $\mathcal{D}$, and we say $A \prec B$ if $A \preceq B$ and $B \not\preceq A$, i.e., $\rho(A) = \rho(B) \ \wedge\ unres(A) = unres(B) \ \wedge\ res(A) <_p res(B)$. $\square$

We can intuitively argue that "$\prec$" is a sound notion of preemption (Theorem 5.4 is a formal statement of this fact). Assume that a term may execute either A or B (e.g., $(A : E)+(B : F)$). First, if $\rho(A) = \rho(B)$, contexts with resource constraints (such as Close and Parallel) will affect both A and B in the same manner. Next, if $unres(A) = unres(B)$ (that is, the unsynchronized parts of A and B are identical), when the environment offers some event C, it will be able to synchronize with A if and only if it can synchronize with B. Finally, since $res(A)$ and $res(B)$ are hosted on the same resources and are fully synchronized, they will interact in an identical manner with all environments. Thus if $res(A) <_p res(B)$, we may maximize our priority arbitration and choose B over A.

We argue that any sound, nontrivial preemption order *must* make use of the three ingredients represented in Definition 5.1: resource utilization, synchronization, and priority. While omitting one or more of these factors may yield a more straightforward definition for "$\prec$", the result will generate an unsound semantics. In the following examples we show some simple, albeit poor choices for "$\prec$". They are "poor" because they allow us to replace a term $(A : E) + (B : F)$ with $B : F$, without accounting for the influence of certain contexts in which the term may appear. That is, in some contexts we may indeed be able to replace one term for the other, while in others we may not. This deficiency leads to a proof system that is not compositional.

Example 5.1 Assume that "$\prec$" is "greedy" in the following sense. Each resource makes its own preemption decisions, and excludes environmental effects in such decisions; further, any event $a \in \Sigma_i$ such that $\pi(a) > 0$ can preempt idle time on resource i. In other words, $A \prec B$ if and only if $A <_p B$.

Using an example, we can easily show why this preemption order fails in the context of resource sharing. Let $a, b \in \Sigma_1$, with $\pi(a) = 1$ and $\pi(b) = 1$. Let a and b occupy their own connection sets; that is, they are local to resource 1. Now consider the term, in which the two constituent terms must be interleaved on resource 1:

$$((\{a\} : NIL) + (\emptyset : NIL))_{\{1\}}\|_{\{1\}}(\{b\} : NIL)$$

At the first time unit, the left hand side may execute either "$\{a\}$" or "$\emptyset$", while the right hand side must execute "$\{b\}$". However, both a and b are mapped to the same resource, and thus, they may not be simultaneously executed at the first time unit. Since both constituent terms *must* execute some action at time 1, the left hand side must execute "$\emptyset$" to accommodate the execution of "$\{b\}$". In other words, at time 1, "$\{b\}$" is executed when they are composed in parallel.

But according to this preemption order, we have $\emptyset \prec \{a\}$ and thus, $\{a\} : NIL$ can be substituted for $(\{a\} : NIL) + (\emptyset : NIL)$. However, the following term is not capable of executing any action at time 1:

$$(\{a\} : NIL)_{\{1\}}\|_{\{1\}}(\{b\} : NIL)$$

□

Example 5.2 As a modest improvement, assume that $A \prec B$ if and only if both $\rho(A) = \rho(B)$ and $A <_p B$. Let $a!, b \in \Sigma_1$, $a? \in \Sigma_2$, with $\pi(a!) = 1$, $\pi(b) = 2$ and $\pi(a?) = 1$. Further, let there be two connection sets: $\{a?, a!\}$ and $\{b\}$. Consider the following term:

$$((\{a!\} : NIL) + (\{b\} : NIL))_{\{1\}}\|_{\{2\}}(\{a?\} : NIL)$$

Here, synchronization is forced between "$a!$" and "$a?$" and thus, the term may only execute "$\{a!, a?\}$" at the first time unit. But since $\{a!\} \prec \{b\}$, the above term should be equivalent to:

$$(\{b\} : NIL)_{\{1\}}\|_{\{2\}}(\{a?\} : NIL)$$

However, when "$\{a?\}$" is executed by resource 2, it must synchronize with "$\{a!\}$" when combined with resource 1. Because only $\{b\}$ is offered, the term is equivalent to NIL. Since this preemption order neglects all synchronization information, it is clearly a poor choice. □

Example 5.3 To take a safer approach, assume that "$\prec$" is defined as follows:

$$A \prec B \quad \text{if and only if} \quad \rho(A) = \rho(B) \land fullsync(A) \land fullsync(B) \land A <_p B$$

Let $a, b \in \Sigma_1$, $c? \in \Sigma_2$, with $\pi(a) = 1$, $\pi(b) = 2$ and $\pi(c?) = 1$. Assume that $fullsync(\{a\})$ and $fullsync(\{b\})$ both hold; that is, they are local to resource 1. Also, assume that $fullsync(\{c?\})$ does not hold; that is, it is not local to resource 2. But let $connections(c?) \cap \Sigma_1 = \emptyset$. This means that since no connections of "$c?$" reside on resource 1, $\{c?\}$ may be executed as if it were local to resource 2. Now consider the following term:

$$((\{a\} : NIL) + (\{b\} : NIL))_{\{1\}}\|_{\{2\}}(\{c?\} : NIL)$$

Since the left hand side may execute either $\{a\}$ or $\{b\}$, while the right hand side executes $\{c?\}$, at the first time unit we may either observe $\{a, c?\}$ or $\{b, c?\}$. And because neither action is fully synchronized, $\{a, c?\} \not\prec \{b, c?\}$.

However, it is true that $\{a\} \prec \{b\}$, and thus, the following term should exhibit the same behavior as that above:

$$(\{b\} : NIL)_{\{1\}}\|_{\{2\}}(\{c?\} : NIL)$$

But they are not identical, since this term may execute only $\{b, c?\}$ at the first time unit. So we must conclude that again our choice for "$\prec$" was incorrect. □

5.3 Prioritized Transition System

Now we define the transition system $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$, grounded in our notion of preemption.

Definition 5.2 *The labeled transition system $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$ is a relation $\rightarrow_\pi \subseteq \mathcal{E} \times \mathcal{D} \times \mathcal{E}$ and is defined as follows: $(E, A, E') \in \rightarrow_\pi$ (or $E \xrightarrow{A}_\pi E'$) if:*

1. $E \xrightarrow{A} E'$, *and*

2. For all $A' \in \mathcal{D}$, $E'' \in \mathcal{E}$ *such that* $E \xrightarrow{A'} E''$, $A \not\prec A'$. □

Example 5.4 Consider the term $E \overset{\text{def}}{=} (\{a!, a?\} : NIL) + (\{b!, b?\} : NIL)$, where $\rho(\{a!, b!\}) = i$ and $\rho(\{a?, b?\}) = \{j\}$. Now assume that $\pi(a!) < \pi(b!)$, $\pi(a?) < \pi(b?)$, and further, that both $\{a!, a?\}$ and $\{b!, b?\}$ are connection sets. Then $\{a!, a?\} \prec \{b!, b?\}$. While E has two initial transitions under "$\longrightarrow$", there is only one prioritized transition: $E \xrightarrow{\{b!,b?\}}_{\pi} NIL$. □

Example 5.5 Consider the term $E \overset{\text{def}}{=} (\{a\} : NIL) + (\emptyset : E)$, where $\rho(\{a\}) = i$, with $fullsync(\{a\})$ and $\pi(a) = 1$. That is, E is the term that indefinitely idles before executing {a}. And although $\emptyset <_p \{a\}$, it is not true that $\emptyset \prec \{a\}$. As we showed in 5.2, E may be interleaved with another term that must initially use resource i (see 5.1). Thus "$\longrightarrow_{\pi}$" admits two initial actions for E, $\{a\}$ and $\emptyset$.

However, there are also times when we know that E is to be the *sole* resident of resource i. In such a case we want E to initially execute $\{a\}$, since the only other alternative is to idle at a lower priority. To achieve this effect we close E with respect to resource i. Note that under "$\longrightarrow$" there are two potential initial actions – $\{a\}$ and $\{\tau_i^0\}$. But since $\{\tau_i^0\} \prec \{a\}$, there is only one initial action under "$\longrightarrow_{\pi}$," which is $\{a\}$. Thus, when we know that a resource is not going to be utilized further, we can employ the Close operator to increase the degree of preemption. □

Clearly we cannot characterize the prioritized semantics by naively substituting "$\longrightarrow_{\pi}$" for "$\longrightarrow$" in the structured transition rules (Figure 4). If this were the case "$\longrightarrow$" and "$\longrightarrow_{\pi}$" would describe the same relation, which they do not. Definition 5.2 and Example 5.4 adequately serve to illustrate that "$\longrightarrow_{\pi}$" is *properly* contained in "$\longrightarrow$." Nonetheless, the following theorem demonstrates that "$\longrightarrow_{\pi}$" is sufficiently well-behaved, in that any prioritized transition is derived strictly from other prioritized transitions. The proof is quite detailed, and is presented in Appendix A.

Theorem 5.1 *Let* $E, F \in \mathcal{E}$ *such that* $E \xrightarrow{A}_{\pi} E'$. *Then the following properties hold:*

1. $E \xrightarrow{A} E'$.

2. If $F \xrightarrow{B} F'$ *is a premise of a rule that derives* $E \xrightarrow{A} E'$, *then* $F \xrightarrow{A}_{\pi} F'$. □

The next result shows that "$\prec$" is progress-preserving, in that for a given transition $E \xrightarrow{A} E'$, either $E \xrightarrow{A}_{\pi} E'$, or there is some preempting transition $E \xrightarrow{A'} E''$, with

$A \prec A'$, such that $E \xrightarrow{A'}_\pi E''$. This is an important fact, for it demonstrates that preemption, by itself, cannot produce deadlock in a system.

Theorem 5.2 *If there is an $A \in \mathcal{D}$ and $E, E' \in \mathcal{E}$ such that $E \xrightarrow{A} E'$, then there exist $A' \in \mathcal{D}$, $E'' \in \mathcal{E}$ such that $E \xrightarrow{A'}_\pi E''$ with $A \preceq A'$.*

Proof: Assume that the conclusion is false. Then, setting $A_0 = A$ and inductively applying Definition 5.2, we see that $\forall\, i \in \boldsymbol{N}, i > 0$, there exist $A_i \in \mathcal{D}$, $E_i' \in \mathcal{E}$ such that $E \xrightarrow{A_i} E_i'$ with $A_{i-1} \prec A_i$. So we have the infinite chain over $\mathcal{D}$: $A_0 \prec A_1 \prec A_2 \prec \ldots$, and by Definition 5.1,

$$res(A_0) <_p res(A_1) <_p res(A_2) <_p \ldots$$

However, note that $\forall i, j \in \boldsymbol{N}$, $\rho(A_i) = \rho(A_j)$, and thus $\forall i, j \in \boldsymbol{N}$, $\rho(res(A_i)) = \rho(res(A_j))$. Further, since every $A \in \mathcal{D}$ is finite, $\rho(res(A))$ is finite. Thus there are finitely many distinct priorities on sets using the resources in $\rho(res(A))$: $|\rho(res(A))|(mp + 1)$ to be exact. So such infinite, strictly increasing chains cannot exist. □

5.4 Bisimulation and Priority Equivalence

Equivalence between processes is based on the concept of *strong bisimulation* (Park, 1981), which is defined as follows:

Definition 5.3 *For a given transition system $\langle \mathcal{E}, \rightsquigarrow, \mathcal{D} \rangle$, the symmetric relation $r \subseteq (\mathcal{P}, \mathcal{P})$ is a* strong bisimulation *if, for $(P, Q) \in r$ and $A \in \mathcal{D}$,*

1. *if $P \overset{A}{\rightsquigarrow} P'$ then, for some Q', $Q \overset{A}{\rightsquigarrow} Q'$ and $(P', Q') \in r$, and*
2. *if $Q \overset{A}{\rightsquigarrow} Q'$ then, for some P', $P \overset{A}{\rightsquigarrow} P'$ and $(P', Q') \in r$.* □

We let "$\sim$" denote *unconstrained strong equivalence*, or the largest such bisimulation with respect to the transition system $\langle \mathcal{E}, \rightarrow, \mathcal{D} \rangle$. Relying on the well-known theory found in Milner (1989), "$\sim$" exists, is an equivalence relation over $\mathcal{P}$, and is a congruence with respect to the CCSR operators. Similarly, we denote "$\sim_\pi$" as the largest strong bisimulation over the transition system $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$, and we call it *prioritized strong equivalence*. Again we can state without proof that "$\sim_\pi$" exists, and that it is an equivalence relation over $\mathcal{P}$.

It should be apparent that "$\sim_\pi$" defines a coarser equivalence than "$\sim$". First, the preemptive nature of "$\rightarrow_\pi$" ensures that the two relations are not identical, for example, consider the term E in Example 5.4. While $E \sim_\pi \{b!, b?\} : NIL$, the equivalence certainly does not hold under "$\sim$". But as the next theorem shows, any distinction made by "$\sim_\pi$" will be preserved by "$\sim$".

Theorem 5.3 *Let $P, Q \in \mathcal{P}$ and assume P is strongly equivalent to Q under the transition system $\langle \mathcal{E}, \rightarrow, \mathcal{D} \rangle$, (that is, $P \sim Q$). Then $P \sim_\pi Q$.*

Proof: We need only show that the relation "$\sim$" is a bisimulation on the transition system $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$. Assume $P \sim Q$, and let $P \xrightarrow{A}_\pi P'$. By Definition 5.2, $P \xrightarrow{A} P'$. Since $P \sim Q$, there is some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{A} Q'$ with $P' \sim Q'$. Thus we must prove that $Q \xrightarrow{A}_\pi Q'$. If this is false, there is some $A' \in \mathcal{D}$, $Q'' \in \mathcal{P}$ such that $Q \xrightarrow{A'} Q''$ and $A \prec A'$. But since $P \sim Q$, there is also some $P'' \in \mathcal{P}$ such that $P \xrightarrow{A'} P''$, which is a contradiction. Similarly, if $Q \xrightarrow{A}_\pi Q'$, then for some P', $P \xrightarrow{A}_\pi P'$ with $P' \sim Q'$. □

Note that Definition 5.3 gives meaning to "$\sim_\pi$" for the domain of agents. However, using the standard technique, we can easily extend "$\sim_\pi$" to terms with free variables.

Definition 5.4 *For terms E and F, let $free(E) \subseteq \{X_1, \ldots, X_n\}$ and $free(F) \subseteq \{X_1, \ldots, X_n\}$. Then $E \sim_\pi F$ if, for all agents $P_1, \ldots, P_n \in \mathcal{P}$, $E[P_1/X_1, \ldots, P_n/X_n] \sim_\pi F[P_1/X_1, \ldots, P_n/X_n]$.* □

The next theorem states that "$\sim_\pi$" forms a congruence over the CCSR operators.

Theorem 5.4 *Prioritized strong equivalence is a congruence with respect to the CCSR operators. That is, if $E \sim_\pi F$, we have:*

(1) $A : E \sim_\pi A : F$

(2a) $E + G \sim_\pi F + G$ (2b) $G + E \sim_\pi G + F$

(3a) $E_I\|_J G \sim_\pi F_I\|_J G$ (3b) $G_I\|_J E \sim_\pi G_I\|_J F$

(4a) $E \, \Delta^B_t \, (G_1, G_2, G_3) \sim_\pi F \, \Delta^B_t \, (G_1, G_2, G_3)$

(4b) $G_1 \, \Delta^B_t \, (E, G_2, G_3) \sim_\pi G_1 \, \Delta^B_t \, (F, G_2, G_3)$

(4c) $G_2 \, \Delta^B_t \, (G_1, E, G_3) \sim_\pi G_2 \, \Delta^B_t \, (G_1, F, G_3)$

(4d) $G_3 \, \Delta^B_t \, (G_1, G_2, E) \sim_\pi G_3 \, \Delta^B_t \, (G_1, G_2, F)$

(5) $E \backslash A \sim_\pi F \backslash A$

(6) $[E]_I \sim_\pi [F]_I$

(7) $fix(X.E) \sim_\pi fix(X.F)$

Proof: It suffices to prove cases (1)-(6) for agents; Definition 5.4 makes the generalization to terms straightforward. In Appendix B we present the proofs for cases (1), (2a), (3a), (4a), (5) and (6); case (2b) is identical to (2a), case (3b) is identical to (3a), and cases (4b)-(4d) are similar to case (4a). In these proofs we often make use of the fact that "$\sim_\pi$" is the largest prioritized bisimulation over "$\rightarrow_\pi$". Thus, to show that $P_1 \sim_\pi P_2$, it suffices to establish *any* bisimulation r such that $(P_1, P_2) \in r$. Since $r \subseteq \sim_\pi$, $P_1 \sim_\pi P_2$.

As for case (7), we limit ourselves to proving equivalence for terms where $free(E) \cup free(F) \subseteq \{X\}$. To show that $fix(X.E) \sim_\pi fix(X.F)$ we establish their bisimilarity up to "$\sim_\pi$", and we use the standard technique of transition induction. Again, the details of the proof are in Appendix B. □

We now turn briefly to the existence and uniqueness of recursive terms.

Theorem 5.5 *For any term E, $fix(X.E) \sim_\pi E[fix(X.E)/X]$; that is, $fix(X.E)$ satisfies the recursive equation $X \sim_\pi E$.*

Proof: By Theorem 5.3, it suffices to show that $fix(X.E) \sim E[fix(X.E)/X]$. But this result follows directly from the definition of Recursion, since $fix(X.E) \xrightarrow{A} E'$ if and only if $E[fix(X.E)/X] \xrightarrow{A} E'$. □

Theorem 5.6 *Let E be a term such that X is guarded in E. Then for $F, G \in \mathcal{E}$, if $F \sim_\pi E[F/X]$ and $G \sim_\pi E[G/X]$, then $F \sim_\pi G$.*

Proof: As in the proof of Theorem 5.4(7), we limit ourselves the case where $free(E) \subseteq \{X\}$, and establish that $r = \{(H[F/X], H[G/X]) \mid free(H) \subseteq \{X\}\}$ is a bisimulation up to "$\sim_\pi$." The details of the proof are similar to that of Theorem 5.4(7), and we omit them here. □

6 An Axiomatization of CCSR

The axioms in the CCSR proof system, $\mathcal{A}$, are enumerated in Figure 5. We claim that $\mathcal{A}$, (augmented with standard laws for substitution), is sound with respect to prioritized equivalence.

Theorem 6.1 *For any terms $E, F \in \mathcal{E}$, if $\mathcal{A} \vdash E = F$, then $P \sim_\pi Q$.*

Proof: It suffices to present proofs for agents, since equality is preserved by substitution. So for each axiom $P = Q$ in $\mathcal{A}$, we construct a bisimulation to show that $P \sim_\pi Q$. For selected cases, see Appendix C. □

Note that the language is fully distributive, in that all of the operators distribute over Choice. Using this fact we can derive the Expansion Law, that serves to unite several of the Choice and Parallel laws. Let I be an index set representing terms, such that for each $i \in I$, there is some corresponding term E_i. If $I = \{i_1, \ldots, i_n\}$, because of Choice(4) we are able to neglect parentheses and use the following notation:

$$\sum_{i \in I} E_i \stackrel{\text{def}}{=} E_{i_1} + \ldots + E_{i_n}$$

and where $\sum_{i \in \emptyset} E_i \stackrel{\text{def}}{=} NIL$.

Choice(1)	$E + NIL = E$
Choice(2)	$E + E = E$
Choice(3)	$E + F = F + E$
Choice(4)	$(E + F) + G = E + (F + G)$
Choice(5)	$(A : E) + (B : F) = B : F \quad \text{if } A \prec B$
Par(1)	$E\,_I\|_J\, NIL = NIL$
Par(2)	$E\,_I\|_J\, F = F\,_J\|_I\, E$
Par(3)	$(E\,_I\|_J\, F)\,_{(I\cup J)}\|_K\, G = E\,_I\|_{(J\cup K)}\,(F\,_J\|_K\, G) \quad \text{if } I \cap J = \emptyset,\ J \cap K = \emptyset,\ I \cap K = \emptyset$
Par(4)	$E\,_I\|_J\,(F + G) = (E\,_I\|_J\, F) + (E\,_I\|_J\, G)$
Par(5)	$(A : E)\,_I\|_J\,(B : F) = \begin{cases} (A * B) : (E\,_I\|_J\, F) & \text{if } \rho(A) \subseteq I,\ \rho(B) \subseteq J, \\ & \quad \rho(A) \cap \rho(B) = \emptyset,\ sync_{(I\cup J)}(A * B) \\ NIL & \text{otherwise} \end{cases}$
Scope(1)	$NIL\, \Delta_t^B\, (F, G, H) = H$
Scope(2)	$(E_1 + E_2)\, \Delta_t^B\, (F, G, H) = (E_1\, \Delta_t^B\, (F, G, H)) \ + \ (E_2\, \Delta_t^B\, (F, G, H))$
Scope(3)	$(A : E)\, \Delta_t^B\, (F, G, H) = \begin{cases} (A * B : F) + H & \text{if } \surd \in A \\ (A : (E\, \Delta_{t-1}^B\, (F, G, H))) + H & \text{if } \surd \notin A \text{ and } t > 1 \\ (A : G) + H & \text{otherwise} \end{cases}$
Close(1)	$[NIL]_I = NIL$
Close(2)	$[E + F]_I = [E]_I + [F]_I$
Close(3)	$[A : E]_I = \begin{cases} (A \cup (\mathcal{T}_I^0 - \mathcal{T}_{\rho(A)}^0)) : [E]_I & \text{if } \rho(A) \subseteq I \\ NIL & \text{otherwise} \end{cases}$
Close(4)	$[[E]_I]_J = \begin{cases} [E]_J & \text{if } I \subseteq J \\ NIL & \text{otherwise} \end{cases}$
Hide(1)	$NIL \backslash B = NIL$
Hide(2)	$(E + F)\backslash B \ = \ E\backslash B + F\backslash B$
Hide(3)	$(A : E)\backslash B = \begin{cases} \phi_{hide(B)}(A) : (E\backslash B) & \text{if } fullsync(A \cap B) \\ NIL & \text{otherwise} \end{cases}$

Figure 5: The Axiom System, $\mathcal{A}$

Theorem 6.2 (Expansion Law) *Let K and L be finite index sets such that for all $k \in K$, $l \in L$, $A_k : E_k \in \mathcal{E}$ and $B_l : F_l \in \mathcal{E}$. Then*

$$\mathcal{A} \vdash (\sum_{k \in K} A_k : E_k)_I \|_J (\sum_{l \in L} B_l : F_l) \;=\; \sum_{\substack{k \in K,\ l \in L, \\ \rho(A) \subseteq I,\ \rho(A) \cap \rho(B) = \emptyset, \\ \rho(B) \subseteq J,\ sync_{(I \cup J)}(A * B)}} (A_k * B_l) : (E_k\, {}_I\|_J\, F_l)$$

Proof: Immediate from Choice(1),(4) and Par(1),(4),(5). □

Theorem 6.3 *For any finite agents $P, Q \in \mathcal{P}$, if $P \sim_\pi Q$, then $\mathcal{A} \vdash P = Q$.*

Proof: Using $\mathcal{A}$, P and Q can be transformed into $\hat{P}$ and $\hat{Q}$, respectively, where both $\hat{P}$ and $\hat{Q}$ are in *prioritized normal form* (PNF). A term R is in PNF if $R \equiv \sum_{k \in K} C_k : R_k$, where 1) for all $k, l \in K$, $C_k \not\prec C_l$, and 2) each R_k is in PNF. We note that Choice(5) is the key to enforcing property 1); that is, whenever there are $k, l \in K$ with $C_k \prec C_l$, we may invoke the law to eliminate $C_k : R_k$.

So assume that $\hat{P} \equiv \sum_{i \in I} A_i : P_i$ and $\hat{Q} \equiv \sum_{j \in J} B_j : Q_j$. The remainder of the proof follows by induction on the maximum depth of $\hat{P}$ and $\hat{Q}$. If the maximum depth is 0 then $\hat{P} \equiv \hat{Q} \equiv NIL$, and we are done. Otherwise, if $\hat{P} \xrightarrow{A}_\pi P'$, then for some $i \in I$, $A : P' \equiv A_i : P_i$. And since $\hat{P} \sim_\pi \hat{Q}$, $\hat{Q} \xrightarrow{A}_\pi Q'$. So for some $j \in J$, $A : Q' \equiv B_j : Q_j$. Further, $P_i \sim_\pi Q_j$, so by induction, $\mathcal{A} \vdash P_i = Q_j$; thus, $\mathcal{A} \vdash A_i : P_i = B_j : Q_j$. So for all $i \in I$, there is some $j \in J$ such that $\mathcal{A} \vdash A_i : P_i = B_j : Q_j$, and by a similar argument the converse is true as well. So by using Choice(2) to eliminate redundancies, and Choice(3) to regroup terms, it follows that $\mathcal{A} \vdash \hat{P} = \hat{Q}$. □

7 Example, Revisited

In this section we use our proof rules to demonstrate the correctness of the example from Section 4. In Figure 6 we again show the producer-consumer system, along with some auxiliary definitions which simplify the proof.

Our objective is to use the axiom system to prove the following:

$$\mathcal{A} \vdash System \;=\; \{p_1, p_2, \tau_3^0\} : T$$
$$\mathcal{A} \vdash T \;=\; [\ \{int_1!, int_1?\} : \{c_1\} : \{c_1\} : \{int_2!, int_2?\} : \{c_2\} : \{p_1, p_2, c_2\} : T\ ]_{\{1,2,3\}}$$

We present a sketch of the proof in Figure 7. Steps (S1)-(S4) are derived by Scope(2) and Scope(3). In particular we take advantage of the fact that Scope distributes over Choice, as characterized by Scope(2). Using these results, we derive step (S5) by Theorem 6.2, Close(2)

$$\begin{aligned}
Producer_1 &\overset{\text{def}}{=} (\delta_\infty(\{p_1\} : \delta_2(\{int_1!\} : IDLE)))\ \Delta^{\bullet}_6\ (NIL, Producer_1, NIL) \\
Producer_2 &\overset{\text{def}}{=} (\delta_\infty(\{p_2\} : \delta_4(\{int_2!\} : IDLE)))\ \Delta^{\bullet}_6\ (NIL, Producer_2, NIL) \\
Consumer_1 &\overset{\text{def}}{=} \delta_\infty(\{int_1?\} : \delta_\infty(\{c_1\} : \delta_\infty(\{c_1\} : Consumer_1))) \\
Consumer_2 &\overset{\text{def}}{=} \delta_\infty(\{int_2?\} : \delta_\infty(\{c_2\} : \delta_\infty(\{c_2\} : Consumer_2))) \\
System &\overset{\text{def}}{=} [\ (Producer_1\, {}_{\{1\}}\|_{\{2\}}\, Producer_2)\, {}_{\{1,2\}}\|_{\{3\}} \\
&\qquad (Consumer_1\, {}_{\{3\}}\|_{\{3\}}\, Consumer_2)\ ]_{\{1,2,3\}} \\
P_1' &\overset{\text{def}}{=} (\delta_2(\{int_1!\} : IDLE))\ \Delta^{\bullet}_5\ (NIL, Producer_1, NIL) \\
P_1'' &\overset{\text{def}}{=} (\delta_\infty(\{p_1\} : \delta_2(\{int_1!\} : IDLE)))\ \Delta^{\bullet}_5\ (NIL, Producer_1, NIL) \\
P_2' &\overset{\text{def}}{=} (\delta_4(\{int_2!\} : IDLE))\ \Delta^{\bullet}_5\ (NIL, Producer_2, NIL) \\
P_2'' &\overset{\text{def}}{=} (\delta_\infty(\{p_2\} : \delta_4(\{int_2!\} : IDLE)))\ \Delta^{\bullet}_5\ (NIL, Producer_2, NIL) \\
C_1' &\overset{\text{def}}{=} \delta_\infty(\{c_1\} : \delta_\infty(\{c_1\} : Consumer_1)) \\
C_2' &\overset{\text{def}}{=} \delta_\infty(\{c_2\} : \delta_\infty(\{c_2\} : Consumer_2)) \\
S &\overset{\text{def}}{=} (P_1'\, {}_{\{1\}}\|_{\{2\}}\, P_2')\, {}_{\{1,2\}}\|_{\{3\}}\, (Consumer_1\, {}_{\{3\}}\|_{\{3\}}\, Consumer_2) \\
T &\overset{\text{def}}{=} [S]_{\{1,2,3\}}
\end{aligned}$$

Figure 6: Producer-Consumer System with Auxiliary Definitions

$$\text{(S1) } \mathit{Producer}_1 = (\{p_1\} : P'_1) + (\emptyset : P''_1)$$

$$\text{(S2) } \mathit{Producer}_2 = (\{p_2\} : P'_2) + (\emptyset : P''_2)$$

$$\text{(S3) } \mathit{Consumer}_1 = (\{\mathit{int}_1?\} : C'_1) + (\emptyset : \mathit{Consumer}_1)$$

$$\text{(S4) } \mathit{Consumer}_2 = (\{\mathit{int}_2?\} : C'_2) + (\emptyset : \mathit{Consumer}_2)$$

$$\begin{aligned}\text{(S5) } \mathit{System} = \; & \{p_1, p_2, \tau_3^0\} : [S]_{\{1,2,3\}} \\ & + \{p_1, \tau_2^0, \tau_3^0\} : [(P'_1 {}_{\{1\}}\|_{\{2\}} P''_2)_{\{1,2\}} \|_{\{3\}} (\mathit{Consumer}_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)\,]_{\{1,2,3\}} \\ & + \{\tau_1^0, p_2, \tau_3^0\} : [(P''_1 {}_{\{1\}}\|_{\{2\}} P'_2)_{\{1,2\}} \|_{\{3\}} (\mathit{Consumer}_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)\,]_{\{1,2,3\}} \\ & + \{\tau_1^0, \tau_2^0, \tau_3^0\} : [(P''_1 {}_{\{1\}}\|_{\{2\}} P''_2)_{\{1,2\}} \|_{\{3\}} (\mathit{Consumer}_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)\,]_{\{1,2,3\}}\end{aligned}$$

$$\text{(S6) } \mathit{System} = \{p_1, p_2, \tau_3^0\} : [S]_{\{1,2,3\}}$$

$$\begin{aligned}\text{(S7) } [S]_{\{1,2,3\}} = \; & \{\mathit{int}_1!, \mathit{int}_1?, \tau_2^0\} : [((\mathit{IDLE}\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_1, \mathit{NIL}))\; {}_{\{1\}}\|_{\{2\}} \\ & \qquad (\delta_3(\{\mathit{int}_2!\} : \mathit{IDLE}))\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_2, \mathit{NIL}))_{\{1,2\}} \|_{\{3\}} \\ & \qquad (C'_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)]_{\{1,2,3\}} \\ & + \{\mathit{int}_2!, \mathit{int}_2?, \tau_1^0\} : [((\delta_1(\{\mathit{int}_1!\} : \mathit{IDLE}))\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_1, \mathit{NIL})\; {}_{\{1\}}\|_{\{2\}} \\ & \qquad \mathit{IDLE}\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_2, \mathit{NIL}))_{\{1,2\}} \|_{\{3\}} \\ & \qquad (\mathit{Consumer}_1 {}_{\{3\}}\|_{\{3\}} C'_2)]_{\{1,2,3\}} \\ & + \quad \{\tau_1^0, \tau_2^0, \tau_3^0\} : [((\delta_1(\{\mathit{int}_1!\} : \mathit{IDLE}))\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_1, \mathit{NIL})_{\{1\}} \|_{\{2\}} \\ & \qquad (\delta_3(\{\mathit{int}_2!\} : \mathit{IDLE}))\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_2, \mathit{NIL}))_{\{1,2\}} \|_{\{3\}} \\ & \qquad (\mathit{Consumer}_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)]_{\{1,2,3\}})\end{aligned}$$

$$\begin{aligned}\text{(S8) } [S]_{\{1,2,3\}} = \; & \{\mathit{int}_1!, \mathit{int}_1?, \tau_2^0\} : [((\mathit{IDLE}\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_1, \mathit{NIL}))\; {}_{\{1\}}\|_{\{2\}} \\ & \qquad (\delta_3(\{\mathit{int}_2!\} : \mathit{IDLE}))\, \Delta_4^{\bullet}\, (\mathit{NIL}, \mathit{Producer}_2, \mathit{NIL}))_{\{1,2\}} \|_{\{3\}} \\ & \qquad (C'_1 {}_{\{3\}}\|_{\{3\}} \mathit{Consumer}_2)]_{\{1,2,3\}}\end{aligned}$$

$$\begin{aligned}\text{(S9) } [S]_{\{1,2,3\}} = \; & \{\mathit{int}_1!, \mathit{int}_1?, \tau_2^0\} : \{c_1, \tau_1^0, \tau_2^0\} : \{c_1, \tau_1^0, \tau_2^0\} \\ & : \{\mathit{int}_2!, \mathit{int}_2?, \tau_1^0\} : \{c_2, \tau_1^0, \tau_2^0\} : \{p_1, p_2, c_2\} : [S]_{\{1,2,3\}}\end{aligned}$$

$$\begin{aligned}\text{(S10) } [S]_{\{1,2,3\}} = \; & \{\mathit{int}_1!, \mathit{int}_1?, \tau_2^0\} : \{c_1, \tau_1^0, \tau_2^0\} : \{c_1, \tau_1^0, \tau_2^0\} \\ & : \{\mathit{int}_2!, \mathit{int}_2?, \tau_1^0\} : \{c_2, \tau_1^0, \tau_2^0\} : \{p_1, p_2, c_2\} : [\; [S]_{\{1,2,3\}}\;]_{\{1,2,3\}}\end{aligned}$$

$$\text{(S11) } T = [\; \{\mathit{int}_1!, \mathit{int}_1?\} : \{c_1\} : \{c_1\} : \{\mathit{int}_2!, \mathit{int}_2?\} : \{c_2\} : \{p_1, p_2, c_2\} : T\;]_{\{1,2,3\}}$$

Figure 7: Sketch of Equivalence Proof

and Close(3). In step (S6) we invoke Choice(5), which allows the first branch of the Choice to preempt the three others.

We arrive at step (S7) in the manner of (S1)-(S5), by applying Scope(2), Scope(3), Theorem 6.2, Close(2) and Close(3). In step (S8) we again apply Choice(5), which allows the first alternative to preempt both the second and the third.

There is a leap between steps (S8) and (S9), in which the *System* is "flattened out." However, the procedure is similar to that used between steps (S1) and (S8), and we omit the intermediate steps for the sake of brevity. Step (S10) is the result of applying Close(4) to $[S]_{\{1,2,3\}}$. Finally step (S11) is derived by six applications of Close(3), as well as by substituting T for $[S]_{\{1,2,3\}}$. By Theorem 5.6,

$$T \sim_\pi fix(X.[\ \{int_1!, int_1?\} : \{c_1\} : \{c_1\} : \{int_2!, int_2?\} : \{c_2\} : \{p_1, p_2, c_2\} : X\ ]_{\{1,2,3\}})$$

and thus, $System \sim_\pi \{p_1, p_2, \tau_3^0\} : T$. This shows that the system is able to meet its production/consumption every cycle.

The importance of preemption elimination (law Choice(5)) cannot be underestimated here. A simple way to illustrate this is to set $\pi(int_1?)$ to 1, and thus to give $int_2?$ the same priority as that of $int_1?$. In this case, the choice in step (S7) between $\{int_1!, int_1?, \tau_2^0\}$ and $\{int_2!, int_2?, \tau_1^0\}$ becomes nondeterministic. And if the branch corresponding to $\{int_2!, int_2?, \tau_1^0\}$ is taken c_2 will execute for 2 time units, during which the execution of $int_1?$ will be blocked. But since the deadline for $int_1!$ will have expired, $Consumer_1$ will not get the opportunity to consume during that period. In fact, we can prove that

$$\begin{aligned}
\mathcal{A} \vdash\ \ System &= \{p_1, p_2, \tau_3^0\} : T' \\
\mathcal{A} \vdash\ \ T' &= [\ \ \{int_1!, int_1?\} : \{c_1\} : \{c_1\} : \{int_2!, int_2?\} : \{c_2\} : \{p_1, p_2, c_2\} : T' \\
&\quad\ \ + \{int_2!, int_2?\} : \{c_2\} : \{c_2\} : \emptyset : \emptyset : \{p_1, p_2\} : T'\ ]_{\{1,2,3\}}
\end{aligned}$$

That is, $Consumer_1$ may starve completely. Of course, in a semantics without preemption (e.g., under the "$\longrightarrow$" transition system), there would be many more such nondeterministic choices; in fact, the system could idle indefinitely.

We end this section with an observation on the complexity of equivalence proofs. As with most proof systems of this type, exponential blow-up is at times unavoidable. This is especially true when manipulating terms where preemption-elimination cannot take place; e.g., in agents where all events have the same priority. However, in priority-intensive systems such as our example, a natural tactic seems to arise: whenever axiom Choice(5) can be used, it should be used.

8 Related Work in the Semantics of Priority

Previous research has, with varying success, treated some issues of the priority problem. There has been a spate of effort directed toward defining models for concurrency based on "maximum parallelism;" e.g., Salwicki and Müldner (1981), Francez, Lehmann, and Pnueli (1984), Janicki *et. al.* (1986) and Koymans *et. al.* (1988). In these models, if processes are ready to communicate, they *will* communicate. Thus maximum parallelism incorporates a rather limited, bi-level priority scheme, where non-idle events always take precedence over idle events, and contention between non-idle events is resolved nondeterministically. These models share a common deficiency, in that they assume unlimited availability of resources: To enforce the constraint of "no unnecessary idling," each process is mapped to its own, dedicated processor. A maximum parallelism semantics could easily be obtained in CCSR, where we would set $\pi(a) = 1$ for every $a \in \Sigma$, retain $\emptyset$ as the action with lowest priority.

In Baeten, Bergstra and Klop (1987), the notion of priority is added to a finite subset of ACP without the presence of τ-events. This is accomplished by the introduction of a partial order over actions, "$>$", as well as a priority operator, "θ." As an example, if $a > b$, then

$$\theta(ax + by + z) = \theta(ax + z).$$

Thus in the parlance of CCSR, we would say that a preempts b. In this light, a "θ-free" agent P would be interpreted under "$\longrightarrow$", whereas $\theta(P)$ would be interpreted using "$\longrightarrow_\pi$". One major difference between this work and CCSR is that preemption is "greedy" in the sense of Example 5.1. That is, in general $\theta(P)\,|\,\theta(Q)$ does not have the same meaning as $\theta(P\,|\,Q)$, where $|$ represents parallel composition. The reason for this fact is that the priority of the synchronous action, "$a|b$," does not depend on the priorities of its two constituent actions, "a" and "b."

An interesting result from Baeten, Bergstra, and Klop is that the axioms needed to characterize θ cannot be added to ACP's unprioritized axiom systems, and remain sound with respect to a ready or failure semantics. Instead, the finer-grained ready-trace semantics is introduced to give meaning to prioritized processes.

A bi-level priority semantics for CCS is treated in Cleaveland and Hennessy (1990), in which events are divided into two subsets: those of low priority (e.g., τ, a, b), and those of high priority (e.g., $\underline{\tau}, \underline{a}, \underline{b}$). Events may synchronize only with inverses of the *same* priority, which limits the range of priorities to a two-element, total order, as opposed to a partial ordering which we treat in CCSR. When synchronization occurs between two unprioritized events (e.g., a and $\overline{a}$), the result is the unprioritized τ. Similarly, when $\underline{a}$ and $\underline{\overline{a}}$ synchronize, the result is $\underline{\tau}$. This $\underline{\tau}$-event is the only preemptive action, which gives rise to a limited version of our Choice(5) law: for any unprioritized a,

$$\underline{\tau}.P \ + \ a.Q = \underline{\tau}.P$$

We note that this treatment can be subsumed by the CCSR model, by (1) assigning each event a priority of either 0 or 1, and (2) ensuring that connected events have the same priority. In fact, the treatment of priority in CCSR can be considered an extension to the work in Cleaveland and Hennessy, whose ideas contributed to the development of our model.

In Camilleri and Winskel (1991), CCS is extended with a prioritized choice operator. Akin to Occam's PRI ALT, this construct selects the input event of highest priority. In the terminology of CCSR, this notion of priority can be described by the following two restrictions: each connection set has two elements, and only one of these elements has a non-zero priority. Also concentrating on Occam, Barrett (1990) provides a prioritized semantics within the context of CSP. He proceeds to show that in certain contexts, the introduction of priority can preclude the necessity for fairness assumptions (this should also be apparent from our Producer-Consumer example). Again, the emphasis in this work is on guards at the receiving end of a channel. In CCSR we treat the more general problem of priority conflicts; e.g., where $\pi(a?) > \pi(b?)$ but $\pi(b!) > \pi(a!)$.

Janicki (1987) gives a prioritized semantics for programs written in the COSY language. A priority ordering on *events* is introduced, "$<$", which has an interpretation similar to that of CCSR's preemption order on *actions*. That is, whenever there is a choice between executing a and b, and when $a < b$, the program defers to b. Given this notion of priority, Janicki demonstrates that inadequacy of a standard partial ordering interpretation for COSY path expressions. For example, while a program may possess an initial execution such as $\{a, c\}$, it does not follow that the same program can initially execute the action $\{c\}$ (i.e., the set of executions may not be prefix-closed). To remedy this problem, Janicki introduces a semantics based on "multiple firing sequences," in which $\{a, c\}$ has a different interpretation from both $\{a\}\{c\}$ and $\{c\}\{a\}$. This semantics adequately captures the notion of priority in COSY, at the cost of introducing finer-grained definition of equivalence. (Note that Baeten, Bergstra and Klop make a similar adjustment to accommodate prioritized behaviors.) Finally, Janicki shows that while the standard, partially-ordered "vector firing sequences" are not adequate in the general case, this semantics is sufficient for verifying some very useful properties; e.g., deadlock-freedom.

Okulicka (1990) attacks this problem in a different manner, by defining a priority relation that maintains a prefix-closed semantics for COSY programs. The technique is based on the decomposition of a priority relation into "elementary" relations, which are subsets of the original relations. The main result is that if each of these subrelations leads to a prefix-closed semantics, the original relation does as well.

It is interesting to note that while CCSR and COSY started from opposite places, both models converged on the interrelationship between time and priority. In the case of CCSR, our early investigations made it clear that an untimed, partially-ordered semantics was insufficient to capture many real-time behaviors. Thus we adapted a discrete-time, step-sequence semantics. Then, to capture the flavor of a real-time scheduler, we introduced a notion of priority to arbitrate between executions such as $\{a\}\{c\}$ and $\{a, c\}$. On the other hand, the introduction of the priority ordering in COSY *mandated* that these executions be given a different interpretation. This distinction led to Janicki's multiple firing sequences, which, in a sense, is a real-time semantics! This subtle interrelationship between priority and time was informally discussed in Lamport (1984), in which it is debated whether there is justification for introducing priority into time-independent contexts, such as those defined by an interleaving semantics.

An alternative to Janicki's approach is taken in Best and Koutny (1992), in which the authors present a Petri net, Σ, with a priority relation, ρ on its transitions. Here, the objective is to preserve the partially ordered semantics, without resolving to a step-sequence solution. Instead of restricting the type of concurrency permitted, the net itself is transformed, by introducing new places, arcs and transitions. In this manner, the anomalies inherent in prioritized interleaving semantics can be avoided.

Finally, Hooman (1991) investigates the issue of shared resources; in this sense, the goals of his work are closely related to ours. His real-time language and computation model derive from the earlier work of Koymans *et. al.* (1988). However, he goes much further, by developing two logics for the purpose of verification. One is a temporal extension to the classical Hoare triple paradigm, while the other is a metric-space extension to temporal logic. In this regard, the proof techniques are quite different from our own, which are based on syntactic transformations.

9 Conclusion

We have presented a real-time, resource-based process algebra called CCSR. The CCSR syntax includes primitive constructs to express essential real-time functionality, among which are timeouts, interrupts, periodic behaviors and exceptions. Further, there is a single parallel operator that can be used to express both interleaving at the resource level, and lock-step parallelism at the system level.

CCSR's proof system derives from a term equivalence based on strong bisimulation, which incorporates a notion of preemption based on priority, synchronization and resource utilization. This prioritized equivalence is also a congruence (Theorem 5.4), which leads to the compositionality of our proof system. Thus we can prove correctness for a real-time system by modularly reasoning about its subsystems, the usefulness of which was shown in Section 7.

This work can serve as a departure point for several areas of research. For example, it may be argued that strong bisimulation yields an equivalence that is too fine-grained; that is, it distinguishes between processes that may, in fact, behave identically in most "reasonable" operating environments. Perhaps there are weaker notions of equivalence that can also adequately characterize both resource constraints and priority. While observational congruence (Milner, 1989) suggests itself as a candidate, it fails to quantify the passage of time in an appropriate manner. Of more help may be a semantics based on a testing preorder (DeNicola and Hennessy, 1983), which would also result in a notion of process containment. Yet it is not immediately clear how a testing equivalence can be extended to accommodate the interaction between priority, resource utilization and synchronization.

Also of interest is a more general axiomatization of the operators; in particular, perhaps $\mathcal{A}$ can be extended to accommodate a limited class of recursion. In (Milner, 1989a), observational congruence is axiomatized for finite state, CCS terms. If a similar technique could be used for CCSR terms, it would significantly enhance the applicability of the proof system.

Acknowledgements

We gratefully acknowledge the anonymous referees, whose comments significantly contributed to the quality of this paper. We are particularly indebted to referee A, who directed us to a substantial body of related work on the semantics of priority.

A Proof of Theorem 5.1

Property 1 is obvious from Definition 5.2; that is, the transition relation "$\longrightarrow_\pi$" is a subset of "$\longrightarrow$". Property 2, however, requires a thorough case analysis over the structure of CCSR terms.

Case 1: $E = E_1 + E_2$ for some $E_1, E_2 \in \mathcal{E}$. Then either ChoiceL or ChoiceR may have been the derivation rule used; that is, either $E_1 \xrightarrow{A} E'$ or $E_2 \xrightarrow{A} E'$. Without loss of generality, assume that the former is true. If we do not have that $E_1 \xrightarrow{A}_\pi E'$, then there is some $A' \in \mathcal{D}$, $E'' \in \mathcal{E}$ such that $E_1 \xrightarrow{A'} E''$, with $A \prec A'$. But then by ChoiceL, $E_1 + E_2 \xrightarrow{A'} E''$, which violates the fact that $E \xrightarrow{A}_\pi E'$.

Case 2: $E = E_1\,{}_I\|_J\,E_2$. Then only the Parallel rule may be used, which implies that E' is of the form $E_1'\,{}_I\|_J\,E_2'$ for some $E_1', E_2' \in \mathcal{D}$. Further, there are $A_1, A_2 \in \mathcal{D}$ such that $A = A_1 * A_2$, $E_1 \xrightarrow{A_1} E_1'$, $E_2 \xrightarrow{A_2} E_2'$, $\rho(A_1) \subseteq I$, $\rho(A_2) \subseteq J$, $\rho(A_1) \cap \rho(A_1) = \emptyset$, $sync_{(I\cup J)}(A_1 * A_2)$.

We must prove that $E_1 \xrightarrow{A_1}_\pi E_1'$ and $E_2 \xrightarrow{A_2}_\pi E_2'$. To the contrary, assume it is false that $E_1 \xrightarrow{A_1}_\pi E_1'$. Then there is a $A_1' \in \mathcal{D}$ and $E_1'' \in \mathcal{E}$ such that $E_1 \xrightarrow{A_1'} E_1''$ with $A_1 \prec A_1'$. By definition 5.1, $\rho(A_1') = \rho(A_1)$, and thus, $\rho(A_1') \subseteq I$; as well as $\rho(A_1') \cap \rho(A_2) = \emptyset$. And since $unres(A_1') = unres(A_1)$ and $sync_{(I\cup J)}(A_1 * A_2)$, we also have that $sync_{(I\cup J)}(A_1' * A_2)$. But these are exactly the side conditions required for $E_1\,{}_I\|_J\,E_2 \xrightarrow{A_1' * A_2} E_1''\,{}_I\|_J\,E_2'$. Now, since $res(A_1') >_p res(A_1)$, we have that

$$\begin{aligned}
res(A_1 * A_2) &=_p res(A_1 \cup A_2) \\
&= res(A_1) \cup res(A_2) \cup res(unres(A_1) \cup unres(A_2)) \\
&<_p res(A_1') \cup res(A_2) \cup res(unres(A_1) \cup unres(A_2)) \\
&= res(A_1') \cup res(A_2) \cup res(unres(A_1') \cup unres(A_2)) \\
&= res(A_1' \cup A_2) \\
&=_p res(A_1' * A_2)
\end{aligned}$$

Also, $\rho(A_1 * A_2) = \rho(A_1' * A_2)$ and $unres(A_1 * A_2) = unres(A_1' * A_2)$, so $(A_1 * A_2) \prec (A_1' * A_2)$. But this contradicts our original assumption. So, $E_1 \xrightarrow{A_1}_\pi E_1'$ and by a similar argument, $E_2 \xrightarrow{A_2}_\pi E_2'$.

Case 3: $E = E_1 \,\Delta_t^B\, (E_2, E_3, E_4)$ for some $E_1, E_2, E_3, E_4 \in \mathcal{E}$. The proof is similar to case 1.

Case 4: $E = F \backslash C$ for some $F \in \mathcal{E}$, $C \in \mathbf{P}(\Sigma)$. Then the Hiding rule is the only one that can derive $F\backslash C \xrightarrow{A} E'$, where E' must be of the form $F'\backslash C$ for some $F' \in \mathcal{E}$. By the premises of the rule, there exist some $B \in \mathcal{D}$ such that $F \xrightarrow{B} F'$, and further, $fullsync(C)$, $fullsync(B \cap C)$ and $A = \phi_{hide(C)}(B)$.

What we must prove is that $F \xrightarrow{B}_\pi F'$. To contradict, assume there exist $B' \in \mathcal{D}$, $F'' \in \mathcal{E}$ such that $B \prec B'$ and $F \xrightarrow{B'} F''$. We shall proceed to show that $F \backslash C \xrightarrow{\phi_{hide(C)}(B')} F'' \backslash C$. Now, we know that $fullsync(B \cap C)$ holds; thus $unres(B) \cap C = \emptyset$. Since by definition 5.1, $unres(B') = unres(B)$, to show $fullsync(B' \cap C)$ holds we only need to show that $fullsync(res(B') \cap C)$ holds. But,

$$\begin{array}{rcll} res(B') \cap C & = & Connections(res(B')) \cap C & (1) \\ & = & (\bigcup_{b \in res(B')} connections(b)) \cap C & (2) \\ & = & \bigcup_{b \in res(B')}(connections(b) \cap C) & (3) \\ & = & \bigcup_{b \in res(B') \wedge b \in C} connections(b) & (4) \\ & = & \bigcup_{b \in res(B') \cap C} connections(b) & (5) \\ & = & Connections(res(B') \cap C) & (6) \end{array}$$

and thus, $fullsync(res(B') \cap C)$ holds. Line (1) follows from the definition of "*res*", line (2) follows directly from the definition of "*Connections*", while line (3) distributes the intersection across the unions. Line (4) uses the fact that connectivity is an equivalence relation, and thus, connection sets are mutually disjoint. Thus, because $C = Connections(C)$, if there is some $b \in res(B')$ such that $connections(b) \cap C \neq \emptyset$, then $connections(b) \subseteq C$. And since $b \in connections(b)$, $b \in (res(B') \cap C$ holds as well. Lines (5) and (6) are simply restatements of line (4).

Now letting $A' = \phi_{hide(C)}(B')$, we have satisfied the conditions for the transition $F \backslash C \xrightarrow{A'} F'' \backslash C$. Since $unres(B) \cap C = \emptyset$, by the definition of $\phi_{hide(C)}$, $unres(A) = unres(B)$. Similarly, $unres(A') = unres(B')$ and thus, $unres(A) = unres(A')$. Finally, since $\phi_{hide(C)}$ preserves both resources and priority, $A \prec A'$. But this contradicts our original assumption.

Case 5: $E = [F]_I$, where $F \in \mathcal{E}$ and $I \subseteq \mathcal{R}$. Then the only action rule that applies is Close, which implies there are $F' \in \mathcal{E}$, $B \in \mathcal{D}$ such that $F \xrightarrow{B} F'$ and $A = B \cup (\mathcal{T}_I^0 - \mathcal{T}_{\rho(B)}^0)$.

We must show that $F \xrightarrow{B}_\pi F'$. To contradict, assume that there exists $B' \in \mathcal{D}$, $F'' \in \mathcal{E}$ such that $B \prec B'$ and $F \xrightarrow{B'} F''$. Then by definition 5.1, $\rho(B) = \rho(B')$, $unres(B) = unres(B')$ and $res(B) <_p res(B')$.

Now since $\rho(B) \subseteq I$ and $\rho(B) = \rho(B')$, we have $\rho(B') \subseteq I$. Thus by the Close rule, $[F]_I \xrightarrow{A'} [F'']_I$, where $A' = B' \cup (\mathcal{T}_I^0 - \mathcal{T}_{\rho(B')}^0)$. We shall proceed to show that $A \prec A'$. First note that $\rho(A) = \rho(A') = I$. Also, since $\mathcal{T}_I^0 - \mathcal{T}_{\rho(B)}^0$ is fully synchronized, $unres(A) = unres(B)$; similarly, $unres(A') = unres(B')$, and thus $unres(A) = unres(A')$. Finally,

$$\begin{array}{l} res(A) = res(B) \cup \mathcal{T}_I^0 - \mathcal{T}_{\rho(B)}^0, \quad \text{and} \\ res(A') = res(B') \cup \mathcal{T}_I^0 - \mathcal{T}_{\rho(B')}^0. \end{array}$$

But because $\mathcal{T}_I^0 - \mathcal{T}_{\rho(B)}^0 = \mathcal{T}_I^0 - \mathcal{T}_{\rho(B')}^0$, and since $res(B) <_p res(B')$, we have shown that $res(A) <_p res(A')$. But this means that $A \prec A'$, contradicting the original assumption that $[F]_I \xrightarrow{A}_\pi [F']_I$. So, $F \xrightarrow{B} F'$.

Case 6: $E = fix(X.F)$ for some $F \in \mathcal{E}$. Then $F[fix(X.F)/X] \xrightarrow{A} E'$ is the sole premise that derives the inference $fix(X.F) \xrightarrow{A} E'$. So assume there are $F' \in \mathcal{E}$, $B \in \mathcal{D}$ such that $A \prec B$ and $F[fix(X.F)/X] \xrightarrow{B} F'$. But then $fix(X.F) \xrightarrow{B} F'$, which contradicts the fact that $fix(X.F) \xrightarrow{A}_\pi E'$. □

B Proof of Theorem 5.4

Proof of Theorem 5.4(1): We claim that

$$r = \{(A : P, A : Q) \mid P \sim_\pi Q\} \cup \sim_\pi$$

is a strong bisimulation on $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$. Since "$\sim_\pi$" is the largest bisimulation, if r satisfies Definition 5.3, then $r \subseteq \sim_\pi$ and thus, $r = \sim_\pi$.

By the definition of "$\sim_\pi$", if $(P, Q) \in \sim_\pi$, they are bisimilar. So assume that there exists $(A : P, A : Q) \in r$ such that $P \sim_\pi Q$. Properties 1 and 2 of Definition 5.3 follow directly from that fact that i) $A : P \xrightarrow{A}_\pi P$ is the only transition possible for $A : P$, ii) $A : Q \xrightarrow{A}_\pi Q$ is the only transition possible for $A : Q$ and iii) $P \sim_\pi Q$. □

Proof of Theorem 5.4(2a): We claim that

$$r = \{(P + R, Q + R) \mid P \sim_\pi Q \wedge R \in \mathcal{P}\} \cup \sim_\pi$$

is a strong bisimulation on $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D} \rangle$.

Trivially, by the definition of "$\sim_\pi$", if $(P, Q) \in \sim_\pi$, they are bisimilar. So assume that there exists $(P + R, Q + R) \in r$ such that $P \sim_\pi Q$. To prove property 1 of definition 5.3, assume

$$(\dagger) \qquad P + R \xrightarrow{A}_\pi S_1$$

By Theorem 5.1, either $P \xrightarrow{A}_\pi S_1$ or $R \xrightarrow{A}_\pi S_1$.

Case 1: Let $P \xrightarrow{A}_\pi S_1$. Since $P \sim_\pi Q$, it follows that there exists some $S_2 \in \mathcal{P}$ such that $Q \xrightarrow{A}_\pi S_2$ with $S_1 \sim_\pi S_2$. Thus we must show that $Q + R \xrightarrow{A}_\pi S_2$, and the proof of property 1 will be complete for this case. For the sake of contradiction, assume there exist $A' \in \mathcal{D}$, $S_3 \in \mathcal{P}$ such that $A \prec A'$ and $Q + R \xrightarrow{A'} S_3$. Since $Q \xrightarrow{A}_\pi S_2$, it is impossible that $Q \xrightarrow{A'} S_3$; thus we must have $R \xrightarrow{A'} S_3$. But then $P + R \xrightarrow{A'} S_3$, which contradicts ($\dagger$).

Case 2: Let $R \xrightarrow{A}_\pi S_1$. This implies, by the ChoiceR rule, that $Q + R \xrightarrow{A} S_1$. So it remains to be shown that $Q + R \xrightarrow{A}_\pi S_1$. On the contrary, assume that there exist $A' \in \mathcal{D}$,

$S_2 \in \mathcal{P}$ such that $A \prec A'$ and $Q + R \xrightarrow{A'} S_2$. Since $R \xrightarrow{A}_\pi S_1$, it is impossible that $R \xrightarrow{A'} S_2$, so it must be true that $Q \xrightarrow{A'} S_2$. By Theorem 5.2, there exist $A'' \in \mathcal{D}$, $S_3 \in \mathcal{P}$ such that $Q \xrightarrow{A''} S_3$ with $A' \preceq A''$; i.e. $A \prec A''$. So, since $P \sim_\pi Q$, there is some $S_4 \in \mathcal{P}$ such that $P \xrightarrow{A''}_\pi S_4$. It follows that $P \xrightarrow{A''} S_4$, and by the ChoiceL rule, $P + R \xrightarrow{A''} S_4$, which contradicts (†).

This completes the proof of property 1 of 5.3. A symmetric argument shows that r adheres to property 2; thus r is a bisimulation. $\square$

Proof of Theorem 5.4(3a): We claim that $r = \{(P_I\|_J R, Q_I\|_J R) \mid P \sim_\pi Q \wedge R \in \mathcal{P}\}$ is a strong bisimulation on $\langle \mathcal{E}, \rightarrow_\pi, \mathcal{D}\rangle$. By definition, $(P_I\|_J R, Q_I\|_J R)$ is in r. To prove that r satisfies property 1 of definition 5.3, assume there exist $P', R' \in \mathcal{P}$, $A \in \mathcal{D}$ such that

$$(\dagger) \qquad P_I\|_J R \xrightarrow{A}_\pi P'_I\|_J R'.$$

It suffices to show that for some Q', $Q_I\|_J R \xrightarrow{A}_\pi Q'_I\|_J R'$ and that $P' \sim_\pi Q'$. Then by Theorem 5.1 we have that $P \xrightarrow{A_1}_\pi P'$ and $R \xrightarrow{A_2}_\pi R'$, where $\rho(A_1) \subseteq I$, $\rho(A_2) \subseteq J$ and $A = A_1 * A_2$.

Now because $P \sim_\pi Q$, we have that $Q \xrightarrow{A_1}_\pi Q'$, with $P' \sim_\pi Q'$. To finish showing that r enjoys property 1 of definition 5.3, we must prove that $Q_I\|_J R \xrightarrow{A}_\pi Q'_I\|_J R'$. Obviously part 1 of definition 5.2 is satisfied, so assume part 2 is violated. That is, assume there is some $A' \in \mathcal{D}$, $Q'', R'' \in \mathcal{P}$ such that $Q_I\|_J R \xrightarrow{A'} Q''_I\|_J R''$ with $A \prec A'$. Then by Theorem 5.2, we know there exist some $A'' \in \mathcal{D}, Q''', R''' \in \mathcal{P}$ such that $Q_I\|_J R \xrightarrow{A''}_\pi Q'''_I\|_J R'''$ with $A' \preceq A''$, and hence $A \prec A''$.

Again invoking Theorem 5.1, there are A''_1, A''_2 such that $A'' = A''_1 * A''_2$ with $Q \xrightarrow{A''_1}_\pi Q'''$, and $R \xrightarrow{A''_2}_\pi R'''$. And since $P \sim_\pi Q$, there is also some $P''' \in \mathcal{P}$ such that $P \xrightarrow{A''_1} P'''$. But this implies that $P_I\|_J R \xrightarrow{A''} P'''_I\|_J R'''$ with $A \prec A''$, again contradicting our assumption (†). So $Q_I\|_J R \xrightarrow{A}_\pi Q'_I\|_J R'$ and the proof of property 1 is complete. By a symmetric argument, r satisfies property 2 in definition 5.3, and so r is a bisimulation. $\square$

Proof of Theorem 5.4(4a): We show that

$$r = \{(P \overset{B}{\Delta_t} (R,S,T), Q \overset{B}{\Delta_t} (R,S,T)) \mid P \sim_\pi Q\} \cup Id_{\mathcal{E}}$$

is a strong bisimulation, where $Id_{\mathcal{E}}$ is the syntactic identity relation.

Assume that $(U_1, U_2) \in r$. Certainly if $(U_1, U_2) \in Id_{\mathcal{E}}$, U_1 and U_2 are bisimilar. So consider the case where $U_1 = P \overset{B}{\Delta_t} (R,S,T)$, $U_2 = Q \overset{B}{\Delta_t} (R,S,T)$, and further, assume that $U_1 \xrightarrow{A}_\pi U'_1$ for some $A \in \mathcal{D}$, $U_1 \in \mathcal{P}$.

Case 1: The ScopeC rule was used to derive the transition. That is, there is some $P' \in \mathcal{P}$ such that $P \xrightarrow{A} P'$, with $t > 1$, $\surd \notin A$, and with $U'_1 = P' \overset{B}{\Delta_{t-1}} (R,S,T)$. By Theorem 5.1, we have that $P \xrightarrow{A}_{\pi} P'$, and since $P \sim_{\pi} Q$, we know there is some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{A}_{\pi} Q'$. This allows us to use the same ScopeC rule to derive that $U_2 \xrightarrow{A} U_2'$, where $U_2' = Q' \,\Delta^{B}_{t-1}\, (R,S,T)$. And since $(U_1', U_2') \in r$, it remains to be shown that the transition is prioritized; i.e., that $U_2 \xrightarrow{A}_{\pi} U_2'$.

However, assume that there is some $A' \in \mathcal{D}$, $U_2'' \in \mathcal{P}$ such that $U_2 \xrightarrow{A'} U_2''$, with $A \prec A'$. Certainly the ScopeC or ScopeT rules could be used to derive this transition, as it would be necessary to have that $Q \xrightarrow{A'} Q''$ for some $Q'' \in \mathcal{P}$. This would be a violation of the fact that $Q \xrightarrow{A}_{\pi} Q'$. If the transition were derived by the ScopeE rule, there would be some $A'' \in \mathcal{D}$, $Q'' \in \mathcal{P}$ such that $Q \xrightarrow{A'} Q''$, and that $A' = A'' * B$. But recall that $B = \emptyset$ or $\{\surd\}$, and thus, $\rho(A'') = \rho(A')$, $unres(A'') = unres(A')$, and $res(A'') =_p res(A')$. And this means that $A \prec A''$, which would again contradict the fact that $Q \xrightarrow{A}_{\pi} Q'$. Finally, if the ScopeI rule were the one to derive the transition, it would imply that there is some $T' \in \mathcal{P}$ such that $T \xrightarrow{A'} T'$. However, this would mean that $U_1 \xrightarrow{A'} T'$, which would be a violation of our initial assumption that $U_1 \xrightarrow{A}_{\pi} U_1'$.

Cases 2-4: The ScopeE, ScopeT, or ScopeI rules were used to derive the transition. The proofs for are all similar to that of Case 1, except for the fact that $(U_1', U_2') \in Id_{\mathcal{E}}$. □

Before proving case (5), we require the following lemma.

Lemma B.1 *Assume there exist* $A \in \mathbf{P}(\Sigma)$, $C \in \mathcal{D}$, $P, P' \in \mathcal{P}$ *such that* $fullsync(A)$, $fullsync(C \cap A)$ *and* $P \xrightarrow{C}_{\pi} P'$. *Then* $P \backslash A \xrightarrow{B}_{\pi} P' \backslash A$, *where* $B = \phi_{hide(A)}(C)$.

Proof: First, since $P \xrightarrow{C}_{\pi} P'$, it follows that $P \xrightarrow{C} P'$. Thus by the Hiding rule, $P \backslash A \xrightarrow{B} P' \backslash A$. To show that B is maximal with respect to "$\prec$", assume there exists some $B' \in \mathcal{D}$, $R \in \mathcal{P}$ such that $B \prec B'$ and $P \backslash A \xrightarrow{B'} R$. Again employing the Hiding rule, we know that R is of the form $P'' \backslash A$ for some $P'' \in \mathcal{P}$. Further, there is some $C' \in \mathcal{D}$ such the $P \xrightarrow{C'} P'$, $fullsync(C' \cap A)$ and $B' = \phi_{hide(A)}(C')$.

Since $\phi_{hide(A)}$ neither shrinks or add resources, and since, by definition 5.1, $\rho(B) = \rho(B')$, we have that $\rho(C) = \rho(C')$. Also by definition 5.1, $unres(B) = unres(B')$. Now, since $fullsync(C \cap A)$, we are guaranteed that $unres(C) \cap A = \emptyset$, and thus by the definition of $\phi_{hide(A)}$, $unres(C) = unres(B)$. Similarly, $unres(C') = unres(B')$, and so, $unres(C) = unres(C')$. Finally, since $\phi_{hide(A)}$ preserves priority, $C \leq_p C'$, which means that $C \prec C'$. But this contradicts the assumptions of the lemma. □

Proof of Theorem 5.4(5): We shall proceed to show that

$$r = \{(P \backslash A, Q \backslash A) \mid P \sim_{\pi} Q\}$$

is a strong, prioritized bisimulation. To prove property 1 of definition 5.3, assume there exist $B \in \mathcal{D}$, $R \in \mathcal{P}$ such that

$$P \backslash A \xrightarrow{B}_{\pi} R$$

By Theorem 5.1, there exist $C \in \mathcal{D}$, $P' \in \mathcal{P}$ such that $R = P' \backslash A$, $B = \phi_{hide(A)}(C)$, with $P \xrightarrow{C}_{\pi} P'$. Further, by the Hiding rule, we have that $fullsync(A)$ and $fullsync(C \cap A)$.

Since $P \sim_{\pi} Q$, there exists some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{C}_{\pi} Q'$ with $P' \sim_{\pi} Q'$. So by Lemma B.1, $Q \backslash A \xrightarrow{B}_{\pi} Q' \backslash A$, and the proof of property 1 is complete. A symmetric argument shows that r satisfies property 2 of definition 5.3. □

Proof of Theorem 5.4(6): We shall show that

$$r = \{([P]_I, [Q]_I) \mid P \sim_{\pi} Q\}$$

is a strong prioritized bisimulation. To show that r satisfies property 1 of definition 5.3, assume that there exist $A \in \mathcal{D}$, $R \in \mathcal{P}$ such that

$$(\dagger) \qquad [P]_I \xrightarrow{A}_{\pi} R$$

Then by Theorem 5.1, there are $B \in \mathcal{D}$, $P' \in \mathcal{P}$ such that $R = [P']_I$, and $A = B \cup (\mathcal{T}_I^0 - \mathcal{T}_{\rho(B)}^0)$. Since $P \sim_{\pi} Q$, there exists some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{B}_{\pi} Q'$ with $P' \sim_{\pi} Q'$. To finish the proof of property 1, we must show that $[Q]_I \xrightarrow{A}_{\pi} [Q']_I$.

Certainly $[Q]_I \xrightarrow{A} [Q']_I$ by the Close rule, so assume A is not maximal. That is, let there be some $A' \in \mathcal{D}$, $R' \in \mathcal{P}$ such that $A \prec A'$ and $[Q]_I \xrightarrow{A'} R'$. By Theorem 5.2, there are $A'' \in \mathcal{D}$, $R'' \in \mathcal{P}$ such that $[Q] \xrightarrow{A''}_{\pi} R''$ with $A' \preceq A''$; i.e., with $A \prec A''$. So, by Theorem 5.1, there exist $B'' \in \mathcal{D}$, $Q'' \in \mathcal{P}$ such that $R'' = [Q'']_I$, $Q \xrightarrow{B''}_{\pi} Q''$, and $A'' = B'' \cup (\mathcal{T}_I^0 - \mathcal{T}_{\rho(B'')}^0)$.

But again, since $P \sim_{\pi} Q$, there is some $P'' \in \mathcal{P}$ such that $P \xrightarrow{B''}_{\pi} P''$. Since this implies that $P \xrightarrow{B''} P''$, by the Close rule, $[P]_I \xrightarrow{A''} [P'']_I$, which contradicts ($\dagger$). So the proof of property 1 of definition 5.3 is complete, and property 2 is proved by symmetrically. □

Proof of Theorem 5.4(7): We show that

$$r = \{(G[fix(X.E)/X], G[fix(X.F)/X]) \mid free(G) \subseteq \{X\}\}$$

is a prioritized bisimulation up to "$\sim_{\pi}$". The result then follows by taking $G \equiv X$. So we prove by induction on transition inference that

1. Whenever $G[fix(X.E)/X] \xrightarrow{A}_{\pi} P'$, $\exists Q', Q'' \in \mathcal{P}$ such that $G[fix(X.F)/X] \xrightarrow{A}_{\pi} Q''$ with $Q' \sim_{\pi} Q''$ and $(P', Q') \in r$.

2. Whenever $G[fix(X.F)/X] \xrightarrow{A}_\pi Q'$, $\exists P', P'' \in \mathcal{P}$ such that $G[fix(X.E)/X] \xrightarrow{A}_\pi P''$ with $P' \sim_\pi P''$ and $(P', Q') \in r$.

To prove that direction 1 holds, assume that $(G[fix(X.E)/X], G[fix(X.F)/X]) \in r$ and further, that $G[fix(X.E)/X] \xrightarrow{A}_\pi P'$. We perform a case analysis on some possible forms that G may take (we omit the simpler cases).

Case 1: $G \equiv X$. So, $fix(X.E) \xrightarrow{A}_\pi P'$. So by the Recursion rule and Theorem 5.1, $E[fix(X.E)/X] \xrightarrow{A}_\pi P'$. Since this is a shorter inference, by induction we have that $\exists Q', Q''$ such that $E[fix(X.F)/X] \xrightarrow{A}_\pi Q''$, with $Q' \sim_\pi Q''$ and $(P', Q') \in r$. But since $E \sim_\pi F$, by Definition 5.4 $\exists Q'''$ such that $F[fix(X.F)/X] \xrightarrow{A}_\pi Q'''$, with $Q''' \sim_\pi Q''$. Thus $Q''' \sim_\pi Q'$. Then by again employing the Recursion rule, $fix(X.F) \xrightarrow{A} Q'''$. But $fix(X.F) \xrightarrow{A}_\pi Q'''$ as well, since any derivative of $fix(X.F)$ is also a derivative of $F[fix(X.F)/X]$.

Case 2: $G \equiv G_1\, {}_I\|_J\, G_2$. Since substitution distributes over Parallel composition, we have that $G_1[fix(X.E)/X]\, {}_I\|_J\, G_2[fix(X.E)/X] \xrightarrow{A}_\pi P'$. By the Parallel rule and Theorem 5.1, there are $A_1, A_2 \in \mathcal{D}$, $P_1', P_2' \in \mathcal{P}$ such that $G_1[fix(X.E)/X] \xrightarrow{A_1}_\pi P_1'$, $G_2[fix(X.E)/X] \xrightarrow{A_2}_\pi P_2'$, with $A = A_1 * A_2$ and $P' = P_1'\, {}_I\|_J\, P_2'$. Further, A_1 and A_2 obey the side conditions of the rule.

Since both of these transitions are derived by shorter inference, by induction we have Q_1', Q_1'', Q_2', Q_2'' such that

$$\begin{array}{lll} G_1[fix(X.F)/X] \xrightarrow{A_1}_\pi Q_1'' & Q_1' \sim_\pi Q_1'' & (P_1', Q_1') \in r \\ G_2[fix(X.F)/X] \xrightarrow{A_2}_\pi Q_2'' & Q_2' \sim_\pi Q_2'' & (P_2', Q_2') \in r \end{array}$$

So, $G_1[fix(X.F)/X]\, {}_I\|_J\, G_2[fix(X.F)/X] \xrightarrow{A} Q_1''\, {}_I\|_J\, Q_2''$. Also, since parallel composition preserves "$\sim_\pi$" (parts (3a)-(3b) above), we have that $Q_1''\, {}_I\|_J\, Q_2'' \sim_\pi Q_1'\, {}_I\|_J\, Q_2'$. And since substitution distributes over composition, it follows that $(P_1'\, {}_I\|_J\, P_2', Q_1'\, {}_I\|_J\, Q_2') \in r$.

So we must show that $G_1[fix(X.F)/X]\ {}_I\|_J\ G_2[fix(X.F)/X] \xrightarrow{A}_\pi Q_1''\, {}_I\|_J\, Q_2''$. Assume the contrary, i.e., there are $A' \in \mathcal{D}$, $Q_1''', Q_2''' \in \mathcal{P}$ such that

$$G_1[fix(X.F)/X]\, {}_I\|_J\, G_2[fix(X.F)/X] \xrightarrow{A'} Q_1'''\, {}_I\|_J\, Q_2'''$$

with $A \prec A'$. Then by Theorem 5.2, we know there exist some $A'' \in \mathcal{D}, Q_1'''', Q_2'''' \in \mathcal{P}$ such that $G_1[fix(X.F)/X]\, {}_I\|_J\, G_2[fix(X.F)/X] \xrightarrow{A''}_\pi Q_1''''\, {}_I\|_J\, Q_2''''$, with $A \prec A' \prec A''$. Then by the Parallel rule and Theorem 5.1, there are $A_1'', A_2'' \in \mathcal{D}$ such that $G_1[fix(X.F)/X] \xrightarrow{A_1''}_\pi Q_1''''$, $G_2[fix(X.F)/X] \xrightarrow{A_2''}_\pi Q_2''''$, with $A'' = A_1'' * A_2''$.

Further, since these transitions are derived by shorter inference we can invoke the induction hypothesis to claim there are $P_1'', P_1''', P_2'', P_2'''$

$$G_1[fix(X.E)/X] \xrightarrow{A_1''}_\pi P_1'' \quad P_1'' \sim_\pi P_1''' \quad (P_1''', Q_1'''') \in r$$
$$G_2[fix(X.E)/X] \xrightarrow{A_2''}_\pi P_2'' \quad P_2'' \sim_\pi P_2''' \quad (P_2''', Q_2'''') \in r$$

But then by the Parallel rule, $G_1[fix(X.E)/X]_I\|_J G_2[fix(X.E)/X] \xrightarrow{A''} P_1''{}_I\|_J P_2''$; but since $A \prec A''$, this contradicts our original assumption.
Case 3: $G \equiv fix(Y.H)$, where $Y \not\equiv X$. So $(fix(Y.H))[fix(X.E)/X] \xrightarrow{A}_\pi P'$. Since Y does not occur in E, we manipulate substitutions to get that

$$fix(Y.(H[fix(X.E)/X])) \xrightarrow{A}_\pi P'.$$

Then by the Recursion rule and Theorem 5.1, we have that

$$(H[fix(X.E)/X])[fix(Y.(H[fix(X.E)/X]))/Y] \xrightarrow{A}_\pi P'.$$

Again rearranging substitutions, it follows that $(H[fix(Y.H)/Y])[fix(X.E)/X] \xrightarrow{A}_\pi P'$. So employing the induction hypothesis, there are Q', Q'' such that

$$(H[fix(Y.H)/Y])[fix(X.F)/X] \xrightarrow{A}_\pi Q''$$

with $Q' \sim_\pi Q''$ and $(P', Q') \in r$. Manipulating substitutions in the opposite direction, we have

$$(H[fix(X.F)/X])[fix(Y.(H[fix(X.F)/X]))/Y] \xrightarrow{A}_\pi Q''.$$

So by Recursion (and an argument similar to case 1), $fix(Y.(H[fix(X.F)/X])) \xrightarrow{A}_\pi Q''$. Finally we manipulate substitutions to derive that $(fix(Y.H))[fix(X.F)/X] \xrightarrow{A}_\pi Q''$. □

C Proof of Theorem 6.1

Within the proof of Theorem 6.1 we make use of the following notation: For any set $A \subseteq \Sigma$, $A^\surd$ means $A \cup \{\surd\}$.

Lemma C.1 *For any $I \in \mathcal{R}$, if $sync_{(I)}(A)$, then for any $I' \subseteq I$, $sync_{(I')}(A \cap \Sigma_{I'}^\surd)$.*
Proof: By the definition of $sync_{(I)}$, $A = Connections(A) \cap \Sigma_I^\surd$. So, since $I' \subseteq I$,

$$A \cap \Sigma_{I'}^\surd = Connections(A) \cap \Sigma_{I'}^\surd.$$

We must show that $A \cap \Sigma^{\surd}_{I'} = \mathit{Connections}(A \cap \Sigma^{\surd}_{I'}) \cap \Sigma^{\surd}_{I'}$. To do so, assume that $a \in A \cap \Sigma^{\surd}_{I'}$. But then $a \in \mathit{Connections}(A \cap \Sigma^{\surd}_{I'})$, and since $a \in \Sigma^{\surd}_{I'}$, $a \in \mathit{Connections}(A \cap \Sigma^{\surd}_{I'}) \cap \Sigma^{\surd}_{I'}$. To prove the "$\supseteq$" part, assume that $a \in \mathit{Connections}(A \cap \Sigma^{\surd}_{I'}) \cap \Sigma^{\surd}_{I'}$. Since

$$\mathit{Connections}(A \cap \Sigma^{\surd}_{I'}) \cap \Sigma^{\surd}_{I'} \subseteq \mathit{Connections}(A) \cap \Sigma^{\surd}_{I'},$$

we have that $a \in \mathit{Connections}(A) \cap \Sigma^{\surd}_{I'}$, and thus, $a \in A \cap \Sigma^{\surd}_{I'}$. □

Proof of Par(3): We claim that

$$\begin{aligned} r = \{((P\,{}_I\|_J\,Q)_{(I\cup J)}\|_K\,R,\ P\,{}_I\|_{(J\cup K)}(Q\,{}_J\|_K\,R)) \mid \\ P,Q,R \in \mathcal{P} \wedge I \cap J = \emptyset,\ J \cap K = \emptyset,\ I \cap K = \emptyset\} \end{aligned}$$

is a strong bisimulation on $\langle \mathcal{E}, \rightarrow_\pi\ \mathcal{D}\rangle$. To prove that r satisfies property 1 of definition 5.3, assume there exist $P', Q', R' \in \mathcal{P}$, $A \in \mathcal{D}$ such that

$$(\dagger) \qquad (P\,{}_I\|_J\,Q)_{(I\cup J)}\|_K\,R \xrightarrow{A}_\pi (P'\,{}_I\|_J\,Q')_{(I\cup J)}\|_K\,R'$$

By the definition of Parallel composition we have that:

$$(\ddagger)\ \ P \xrightarrow{A_I} P', \ \ Q \xrightarrow{A_J} Q', \ \ R \xrightarrow{A_K} R'$$

where $\rho(A_I) \subseteq I$, $\rho(A_J) \subseteq J$, $\rho(A_K) \subseteq K$, and $A = A_I * A_J * A_K$. Also, we know that $\mathit{sync}_{(I\cup J\cup K)}(A)$, and thus by by lemma C.1, $\mathit{sync}_{(J\cup K)}((A_I * A_J * A_K) \cap \Sigma^{\surd}_{J\cup K})$. But since I, J and K are mutually disjoint,

$$(A_I * A_J * A_K) \cap \Sigma^{\surd}_{J\cup K} = \begin{cases} A_J * A_K & \text{if } \surd \in A_I \\ (A_J * A_K) - \{\surd\} & \text{if } \surd \notin A_I \end{cases}$$

It is easy to verify that $\mathit{sync}_{(J\cup K)}(A_I * A_J)$ holds if and only if $\mathit{sync}_{(J\cup K)}((A_I * A_J) - \{\surd\})$ holds. Thus, with ($\ddagger$), we have that $Q\,{}_J\|_K\,R \xrightarrow{A_J * A_K} Q'\,{}_J\|_K\,R'$. Also, with ($\ddagger$), as well as the fact that $\mathit{sync}_{(I\cup J\cup K)}(A)$, we find that:

$$P\,{}_I\|_{(J\cup K)}(Q\,{}_J\|_K\,R) \xrightarrow{A} P'\,{}_I\|_{(J\cup K)}(Q'\,{}_J\|_K\,R')$$

It remains to be shown that $P\,{}_I\|_{(J\cup K)}(Q\,{}_J\|_K\,R) \xrightarrow{A}_\pi P'\,{}_I\|_{(J\cup K)}(Q'\,{}_J\|_K\,R')$. To the contrary, assume there is some $A' \in \mathcal{D}$, $P'', Q'', R'' \in \mathcal{P}$, such that $A \prec A'$ and

$$P\,{}_I\|_{(J\cup K)}(Q\,{}_J\|_K\,R) \xrightarrow{A'} P''\,{}_I\|_{(J\cup K)}(Q''\,{}_J\|_K\,R'').$$

But then, by the same argument as that above,

$$(P\,{}_I\|_J\,Q)_{(I\cup J)}\|_K\,R \xrightarrow{A'} (P''\,{}_I\|_J\,Q'')_{(I\cup J)}\|_K\,R'',$$

which contradicts (†). Finally, since

$$((P'{}_I\|_J Q')_{(I\cup J)}\|_K R', P'{}_I\|_{(J\cup K)}(Q'{}_J\|_K R')) \in r,$$

the proof of property 1 is complete. By a symmetric argument, r satisfies property 2 in definition 5.3, and so r is a bisimulation. □

Proof of Par(4): We claim that

$$r = \{(P_I\|_J(Q+R), (P_I\|_J R) + (Q_I\|_J R)) \mid P, Q, R \in \mathcal{P}\} \cup Id_{\mathcal{E}}$$

is a strong, prioritized bisimulation, where $Id_{\mathcal{E}}$ is the identity relation on CCSR terms. Obviously $Id_{\mathcal{E}}$ is such a bisimulation; thus to show that r satisfies property 1 of Definition 5.3, assume there exist $P', S \in \mathcal{P}$, $A \in \mathcal{D}$ such that

$$(\dagger) \qquad P_I\|_J(Q+R) \xrightarrow{A}_{\pi} P'{}_I\|_J S$$

Then there are $A_1 \subseteq \Sigma_I$, $A_2 \subseteq \Sigma_J$ such that $P \xrightarrow{A_1}_{\pi} P'$, $Q + R \xrightarrow{A_2} S$, and $A = A_1 * A_2$. So by the Choice rules, either $Q \xrightarrow{A_2} S$ or $R \xrightarrow{A_2} S$.

Without loss of generality, assume the former case. By applying the Parallel rule, we have $P_I\|_J Q \xrightarrow{A} P'{}_I\|_J S$, and by applying the ChoiceL rule,

$$(P_I\|_J Q) + (P_I\|_J R) \xrightarrow{A} P'{}_I\|_J S$$

So we must show that $(P_I\|_J Q) + (P_I\|_J R) \xrightarrow{A}_{\pi} P'{}_I\|_J S$. To contradict, assume there are $P'', S' \in \mathcal{P}$, $A' \in \mathcal{D}$ such that $A \prec A'$ and

$$(P_I\|_J Q) + (P_I\|_J R) \xrightarrow{A'} P''{}_I\|_J S'.$$

But by the Choice rules, either $P_I\|_J Q \xrightarrow{A'} P''{}_I\|_J S'$ or $P_I\|_J R \xrightarrow{A'} P''{}_I\|_J S'$. Again without loss of generality, assume the latter case. Then there are $A'_1 \subseteq \Sigma_I$, $A'_2 \subseteq \Sigma_J$ such that $P \xrightarrow{A'_1} P''$ and $R \xrightarrow{A'_2} S'$. By the ChoiceR rule, $Q + R \xrightarrow{A'_2} S'$, and thus by the Parallel rule, $P_I\|_J(Q+R) \xrightarrow{A'} P''{}_I\|_J S'$. But this contradicts (†), and so $(P_I\|_J Q) + (P_I\|_J R) \xrightarrow{A}_{\pi} P'{}_I\|_J S$. Since $(P'{}_I\|_J S, P'{}_I\|_J S) \in Id_{\mathcal{E}} \subseteq r$, the proof of property 1 is complete.

The proof of property 2 is similar. □

Proof of Hide(2): We show that

$$r = \{((P+Q)\backslash B, P\backslash B + Q\backslash B) \mid P, Q \in \mathcal{P}, B \in \mathbb{P}(\Sigma)\} \cup Id_{\mathcal{E}}$$

is a strong prioritized bisimulation. We shall prove that r satisfies property 1 of definition 5.3; the proof for property 2 is similar.

Since $Id_{\mathcal{E}}$ is a bisimulation, to show that r satisfies property 1, assume there exist $R \in \mathcal{P}$, $A \in \mathcal{D}$ such that

$$(\dagger) \qquad (P+Q)\backslash B \xrightarrow{A}_{\pi} R$$

By the Hiding rule, we know that R is of the form $S\backslash B$ for some $S \in \mathcal{P}$. Also, there exists some $C \in \mathcal{D}$ such that $P + Q \xrightarrow{C} S$, $\mathit{fullsync}(C \cap B)$ and that $A = \phi_{hide(B)}(C)$. Thus by the Choice rules, either $P \xrightarrow{C} S$ or $Q \xrightarrow{C} S$. Without loss of generality, assume the former case. Then once again applying the Hiding rule, we have that $P\backslash B \xrightarrow{A} S\backslash B$. So by the ChoiceL rule, $P\backslash B + Q\backslash B \xrightarrow{A} S\backslash B$, or equivalently,

$$P\backslash B + Q\backslash B \xrightarrow{A} R$$

We must show that $P\backslash B + Q\backslash B \xrightarrow{A}_{\pi} R$. To contradict, assume there are $A' \in \mathcal{D}$, $R' \in \mathcal{P}$, such that $A \prec A'$ and $P\backslash B + Q\backslash B \xrightarrow{A'} R'$. By the Choice rules, either $P\backslash B \xrightarrow{A'} R'$ or $Q\backslash B \xrightarrow{A'} R'$.

Assume the first case is true (the proof for the second case is equivalent). By the Hiding rule, R' is of the form $S'\backslash B$ for some $S' \in \mathcal{P}$. Further, there is some $C' \in \mathcal{D}$ such that $P \xrightarrow{C'} S'$ with $\mathit{fullsync}(C' \cap B)$ and $A' = \phi_{hide(B)}(C')$. Then by the ChoiceL rule, $P+Q \xrightarrow{C'} S'$, and by the Hiding rule, $(P+Q)\backslash B \xrightarrow{A'} S'\backslash B$. But this contradicts $(\dagger)$, and so, $P\backslash B + Q\backslash B \xrightarrow{A}_{\pi} R$. Since $(R, R) \in r$, the proof of property 1 is complete. $\square$

References

Arnold, A. (1982), Synchronized behaviours of processes and rational relations, *Acta Informatica* **17**, 21–29.

Baeten, J., Bergstra, J., and Klop, J. (1987), Ready-trace semantics for concrete process algebra with a priority operator, *Computer Journal* **30**, 6, 498–506.

Barrett, G. (1990), The semantics of priority and fairness in occam, *in* "Proc. of 5th Int. Conf. Math. Foundations of Programming Semantics," Lecture Notes in Comput. Sci. Vol. **442**, Springer-Verlag, Berlin.

Best, E., and Koutny, M. (1992), Petri net semantics of priority systems, *Theoretical Computer Science* (to appear).

Camilleri, J., and Winskel, G. (1991), CCS with priority choice, *in* "*Proc. of IEEE Symposium on Logic in Computer Science*," IEEE Computer Society Press, Los Alamitos.

Cleaveland, R., and Hennessy, M. (1990), Priorities in process algebras, *Information and Computation* **87**, 58–77.

Francez, N., Lehmann, D., and Pnueli, A. (1984), A linear history semantics for distributed programming, *Theoretical Computer Science* **32**, 25–46.

Gerber, R. (1991), "Communicating Shared Resources: A Model for Distributed Real-Time Systems," PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Hoare, C. (1978), Communicating sequential processes, *Communications of the ACM* **21**, No. 8, 666–676.

Hoare, C. (1985), "Communicating Sequential Processes," Prentice-Hall, Englewood Cliffs.

Hooman, J. (1991), "Specification and Compositional Verification of Real-Time Systems," PhD thesis, Eindhoven University of Technology.

Huizing, C., Gerth, R., and de Roever, W. (1987), Full abstraction of a denotational semantics for real-time concurrency, *in* "Proc. 14^{th} ACM Symposium on Principles of Programming Languages," pp. 223–237, ACM Press, New York.

Janicki, R. (1987), A formal semantics of concurrent systems with a priority relation, *Acta Informatica* **24**, 33–55.

JANICKI, R., LAUER, P., KOUTNY, M., AND DEVILLERS, R. (1986), Concurrent and maximally concurrent evolution of nonsequential systems, *Theoretical Computer Science* **43**, 213–238.

KOYMANS, R., SHYAMASUNDAR, R., DE ROEVER, W., GERTH, R., AND ARUN-KUMAR, S. (1988), Compositional semantics for real-time distributed computing, *Information and Computation* **70**, 210–256.

LAMPORT, L. (1984), What it means for a concurrent program to satisfy a specification: why no one has specified priority, *in* "3rd ACM Symposium on Principles of Distributed Computing," ACM Press, New York.

LEE, I., AND GEHLOT, V. (1985), Language constructs for distributed real-time programming, *in* "Proc. IEEE Real-Time Systems Symposium," IEEE Computer Society Press, Los Alamitos.

LYNCH, N., AND TUTTLE, M. (1988), "An Introduction to Input/Output Automata," Tech. Rep. MIT/LCS/TM-373, Laboratory for Computer Science, Massachusetts Institute of Technology.

MILNER, R. (1980), "A Calculus for Communicating Systems," Lecture Notes in Comput. Sci. Vol. **92**, Springer-Verlag, Berlin.

MILNER, R. (1983), Calculi for synchrony and asynchrony, *Theoretical Computer Science* **25**, 267–310.

MILNER, R. (1989) "Communication and Concurrency," Prentice-Hall, Englewood Cliffs.

MILNER, R. (1989a) A complete axiomatisation for observational congruence of finite-state behaviors, *Information and Computation* **81**, 227–247.

DENICOLA, R., AND HENNESSY, M. (1983), Testing equivalences for processes, *in* "Proc. of Int. Conf. on Automata, Languages and Programming," Lecture Notes in Comput. Sci. Vol. **154**, pp. 548–560, Springer-Verlag, Berlin.

OKULICKA, F. (1990), On priority in cosy, *Theoretical Computer Science* **74**, 199–216.

PARK, D. (1981), Concurrency and automata on infinite sequences, *in* "Proceedings, 5th GI Conference," Lecture Notes in Comput. Sci. Vol. **104**, Springer-Verlag, Berlin.

PLOTKIN, G. (1981), "A Structural Approach to Operational Semantics," Tech. Rep. DAIMI FN-19, Computer Science Dept., Aarhus University.

Salwicki, A., and Müldner, T. (1981), On the algorithmic properties of concurrent programs, *in* "Proceedings of Logic of Programs," Lecture Notes in Comput. Sci. Vol. **125**, Springer-Verlag, Berlin.