An Algebraic Theory of Process Efficiency*

V. Natarajan Rance Cleaveland
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
{nvaidhy@eos, rance@csc}.ncsu.edu

Abstract

This paper presents a testing-based semantic theory for reasoning about the efficiency of concurrent systems as measured in terms of the amount of their internal activity. The semantic preorders are given an algebraic characterization, and their optimality is established by means of a full abstractness result. They are also shown to subsume existing bisimulation-based efficiency preorders. An example is provided to illustrate the utility of this approach.

1. Introduction

Process algebras [12, 10, 3] have attracted substantial attention over the past decade and a half as frameworks for system verification. A hallmark of these theories is the development of *behavioral* equivalences and preorders that relate systems on the basis of their extensional behavior. In these frameworks, specifications are usually given as "high-level" systems; to prove a system meets such a specification, one shows that they are related appropriately by such a behavioral relation. When the behavioral relation is an equivalence, the specification and the implementation have the same external behavior, while if the behavioral relation is a preorder then the implementation is deemed "better" than the specification with respect to some criterion.

Traditional behavioral preorders aim to order processes on the basis of their ability to diverge or behave nondeterministically while being abstract with respect to the degree of internal activity they engage in [4, 18]. Among the most studied of these relations are the testing preorders [15], which relate processes on the basis of their responses to tests. Full operational, denotational and axiomatic characterizations have been developed for these relations, and several case studies attest to their practical utility.

*Research supported by NSF grant CCR-9120995, ONR Young Investigator Award N00014-92-J-1582, NSF Young Investigator Award CCR-9257963, NSF grant CCR-9402807, and AFOSR grant F49620-95-1-0508.

In traditional testing, the only observable behavior a process has involves the external interactions it may have with its environment. While this is adequate for reasoning about the functional behavior of systems, it gives no insight into their relative efficiency. From a practical standpoint, however, efficiency is often an important concern, and developing frameworks that support efficiency-based reasoning would enable existing ad hoc approaches to system efficiency analysis to be formalized. The goal of this paper is to define such a theory using a testing-based paradigm.

The rest of the paper is organized as follows. In the next section we give the syntax and semantics of the language CCS and an overview of the traditional theory of testing. In the following section we introduce a modified notion of testing that supports a determination of the speed with which a process passes a test and develop preorders that relate processes on the basis of their efficiency. This section also establishes the congruence properties and alternative characterizations of these preorders. In Section 4, we demonstrate the naturalness of these preorders by proving they are fully abstract with respect to the notion of providing "faster response". In the section following we compare our work with the existing literature on bisimulation-based efficiency preorders. Section 6 provides a sound and complete axiomatization of these congruences over arbitrary closed terms, while the last two sections contain an illustrative example and our conclusions.

2. CCS and Testing Preorders

This section presents the syntax and operational semantics of the language CCS [12] and an overview of the theory of testing [15].

2.1 Syntax and Semantics of CCS

The language we consider in this paper is taken from [12]. Formally, let Λ denote a countably infinite set of labels; intuitively, Λ contains the "ports" that processes may synchro-

$$\begin{array}{lll}
\alpha.t \xrightarrow{\alpha} t & & \\
s \xrightarrow{\alpha} s' & \Rightarrow & s+t \xrightarrow{\alpha} s', \quad t+s \xrightarrow{\alpha} s' \\
s \xrightarrow{\alpha} s' & \Rightarrow & s|t \xrightarrow{\alpha} s'|t, \quad t|s \xrightarrow{\alpha} t|s' \\
s \xrightarrow{a} s', t \xrightarrow{\overline{a}} t' & \Rightarrow & s|t \xrightarrow{\tau} s'|t' \\
t \xrightarrow{\alpha} t', \alpha \neq \lambda, \overline{\lambda} & \Rightarrow & t\backslash\lambda \xrightarrow{\alpha} t'\backslash\lambda \\
t \xrightarrow{\alpha} t' & \Rightarrow & t[R] \xrightarrow{R(\alpha)} t'[R] \\
t \xrightarrow{\alpha} t' & \Rightarrow & \mathit{fix}(x:t) \xrightarrow{\alpha} t'[\mathit{fix}(x:t)/x]
\end{array}$$

Figure 1. Operational semantics.

nize over. Then the set *Act* of *actions* may be defined by $\mathit{Act} = \Lambda \cup \overline{\Lambda} \cup \{\tau\}$, where $\overline{\Lambda} = \{ \overline{\lambda} \mid \lambda \in \Lambda\}$. Action $\lambda \in \Lambda$ may be thought of as representing the receipt of an input on port λ, while $\overline{\lambda} \in \overline{\Lambda}$ constitutes the depositing of an output on λ; $\tau \notin \Lambda$ represents an internal step. We use $A = \mathit{Act} \setminus \{\tau\}$ to denote the set of all *external* or *visible* actions. In what follows we let $\alpha, \beta, \ldots$ range over *Act* and $a, b, c, \ldots$ over A. We also extend $^{-}$ to all non–τ actions by taking $\overline{\overline{\lambda}} = \lambda$.

The terms of the language may now be defined as follows.

$$t ::= \mathit{nil} \mid \Omega \mid \alpha.t \mid t+t \mid t|t \mid t\backslash\lambda \mid t[R] \mid x \mid \mathit{fix}(x:t)$$

Here R is a *relabeling*, a mapping from *Act* to *Act* preserving τ and $^{-}$. We also assume that relabelings satisfy the following property: if R is a relabeling then the set $\{ a \in A \mid a \neq R(a)\}$ is finite. We use Exp to stand for the set of all terms of the language, and we adopt the usual definitions for *sort* of a term, *free* and *bound* variables, *open* and *closed* terms, *guarded recursion* and *contexts*. We call the closed guarded terms *processes*; $\mathcal{P}$ represents the set of all processes and will be ranged over by $p, q, \ldots$.

The operational semantics of processes is given as a relation $\rightarrow \subseteq \mathcal{P} \times \mathit{Act} \times \mathcal{P}$; we write $p \xrightarrow{\alpha} q$ in lieu of $\langle p, \alpha, q\rangle \in \rightarrow$ and $p \xrightarrow{\alpha}$ if $p \xrightarrow{\alpha} q$ holds for some q. The formal definition is given inductively in Figure 1 and is taken from [12]. Intuitively, $p \xrightarrow{\alpha} q$ holds if p may engage in action α and thereafter behave like q. Accordingly, $\alpha.p$ may engage in α and then behave like p. The operator $+$ constitutes a choice construct: $p + q$ may perform the actions of p or q and then behave like the process from which the action was chosen. Process $p \mid q$ represents the parallel composition of p and q, with concurrent execution modeled via interleaving and synchronization between p and q allowed if they are capable of complementary actions; the result of such a synchronization is a τ. The operator $\backslash\lambda$ defines a scope for actions using the port λ, while $p[R]$ relabels the actions of p in the appropriate manner. $\mathit{fix}(x:t)$ represents a recursively defined process that is a distinguished "solution" to the equation $x = t$.

Definition 2.1 $\downarrow$ *is the smallest predicate on closed terms satisfying the following.*

- $\mathit{nil}\downarrow, \alpha.p\downarrow$.
- $p\downarrow, q\downarrow \Rightarrow (p+q)\downarrow, (p|q)\downarrow, p[R]\downarrow, p\backslash\lambda\downarrow$.
- $t[\mathit{fix}(x:t)/x]\downarrow \Rightarrow \mathit{fix}(x:t)\downarrow$.

Let $p\uparrow$ if it is not the case that $p\downarrow$. Intuitively $p\uparrow$ means that p is an incomplete or partially defined system description.

We end this subsection by explaining some terminology and notation that will be used in this paper. If $s \in \mathit{Act}^*$ then $\hat{s}$ is the sequence obtained by deleting all the occurrences of τ from s. A process p is *unstable* if $p \xrightarrow{\tau}$ and stable otherwise. The set of *extended* integers is the set of integers augmented with two distinguished symbols ∞ and $-\infty$. The usual ordering $\leq$ on integers is also extended to include $x \leq \infty$ and $-\infty \leq x$ where x is any extended integer. If X is a preorder over terms then X^C will stand for the largest precongruence (with respect to CCS) contained in X.

2.2 A Theory of Testing

The traditional testing framework of [6, 15, 8] defines preorders and equivalences that relate processes on the basis of their responses to tests. In this setting a test is any process whose sort contains a distinguished *success* action w; to determine a process' response(s) to a test one must examine the computations that result when the test is "applied" to the process. These intuitions may be formalized as follows.

Definition 2.2

1. *A* configuration *has form $p||t$, where p is a process and t is a test. $p||t$ is* successful *if $t \xrightarrow{w}$. $p||t\downarrow$ if $p\downarrow$ and $t\downarrow$.*
2. *The* configuration transition relation, $\longmapsto$, *is defined by the following rules.*
   - $p \xrightarrow{a} p', t \xrightarrow{a} t' \Rightarrow p||t \overset{a}{\longmapsto} p'||t'$.
   - $p \xrightarrow{\tau} p' \Rightarrow p||t \overset{\tau}{\longmapsto} p'||t$.
   - $t \xrightarrow{\tau} t' \Rightarrow p||t \overset{\tau}{\longmapsto} p||t'$.
3. *A* computation *of $p||t$ is a maximal (potentially infinite) sequence $p||t \equiv p_0||t_0 \xrightarrow{\alpha_1} p_1||t_1 \xrightarrow{\alpha_2} \cdots$; it is* successful *if some $p_i||t_i$ is successful and $p_j||t_j\downarrow$ for all $j < i$. We refer to $\hat{\sigma}$ where $\sigma = \alpha_1.\alpha_2.\cdots$ as the visible content of this computation. The set of all computations of $p||t$ will be denoted by $\mathcal{C}(p,t)$.*
4. *p may t if some computation of $p||t$ is successful. p must t if every computation of $p||t$ is successful.*

As processes and tests are in general nondeterministic, one may distinguish between the possibility (p may t) and the inevitability (p must t) of success. The following preorders may now be defined.

Definition 2.3 $p \sqsubseteq_{may} q$ *if for every test* t *such that* $p\ may\ t$, $q\ may\ t$. $p \sqsubseteq_{must} q$ *if for every test* t *such that* $p\ must\ t$, $q\ must\ t$.

A number of alternative characterizations have been developed to ease the task of reasoning about processes using these preorders. Here we present one due to De Nicola and Hennessy [15].

Definition 2.4 *Let* $s \in A^*$ *be a sequence of visible actions.*

1. $p \stackrel{s}{\Rightarrow} p'$ *is defined inductively as follows:* $p \stackrel{\epsilon}{\Rightarrow} p'$ *if* $p(\stackrel{\tau}{\rightarrow})^* p'$ *and* $p \stackrel{as'}{\Rightarrow} p'$ *if* $p \stackrel{\epsilon}{\Rightarrow}\stackrel{a}{\rightarrow}\stackrel{s'}{\Rightarrow} p'$.
2. *The* language, $L(p)$, *of* p *is defined by:* $L(p) = \{\ s \mid \exists p'.p \stackrel{s}{\Rightarrow} p'\ \}$. *We sometimes refer to the elements of* $L(p)$ *as* traces *of* p.
3. $p\Uparrow$ *iff* $(\exists q :\ p \stackrel{\epsilon}{\Rightarrow} q \wedge q\uparrow)$ *or* $p(\stackrel{\tau}{\rightarrow})^\omega$. $p\Downarrow$ *if* $p\Uparrow$ *is not true.*
4. $p \Downarrow s$ *is defined inductively as follows:* $p\Downarrow \epsilon$ *if* $p\Downarrow$, *and* $p \Downarrow as'$ *if* $p\Downarrow$ *and for all* p' *such that* $p \stackrel{a}{\Rightarrow} p'$, $p' \Downarrow s'$.
5. $S(p) = \{\ a \in A \mid p \stackrel{a}{\Rightarrow}\ \}$.

Intuitively, $L(p)$ contains the sequences of visible actions that p may engage in, while $p\Downarrow$ holds if p cannot engage in an infinite sequence of internal computation steps or evolve autonomously to an under-defined state. $p \Downarrow s$ indicates that during any "execution" of s from p, no state q is reached in which $q\Uparrow$. We now have the following [8].

Theorem 2.5

1. $p \sqsubseteq_{may} q$ *iff* $L(p) \subseteq L(q)$.
2. $p \sqsubseteq_{must} q$ *iff for every* $s \in A^*$ *the following holds.*

$$p \Downarrow s \Rightarrow \begin{cases} q \Downarrow s \\ q \stackrel{s}{\Rightarrow} q' \Rightarrow \exists p' :\ p \stackrel{s}{\Rightarrow} p' \wedge S(p') \subseteq S(q') \end{cases}$$

3. Efficiency Testing

We take the view that a system that performs more internal computation than another in order to exhibit the same external behavior is less efficient than the latter [1]. Accordingly the process $a.\tau.b.nil$ is less efficient than $a.b.nil$. Under traditional testing theory these two processes are deemed equivalent because no test can distinguish between them. This section examines how the traditional testing framework can be modified to make the tests become sensitive to internal activity and how a mathematically tractable theory can be developed for such a framework. We begin with an alteration of the concepts introduced in Definition 2.2.

Definition 3.1 (Efficiency Testing)

1. *A* configuration *has form* $p|_k|t$ *where* p *is a process,* t *is a test and* k *is a natural number.* $p|_k|t$ *is successful if* $t \stackrel{w}{\rightarrow}$. $p|_k|t\downarrow$ *if* $p\downarrow$ *and* $t\downarrow$.
2. *The* configuration transition relation, $\longmapsto$, *is defined by the following rules.*
   - $p \stackrel{a}{\rightarrow} p', t \stackrel{a}{\rightarrow} t', k > 0 \Rightarrow p|_k|t \stackrel{a}{\longmapsto} p'|_{k-1}|t'$.
   - $p \stackrel{\tau}{\rightarrow} p', k > 0 \Rightarrow p|_k|t \stackrel{\tau}{\longmapsto} p'|_{k-1}|t$.
   - $t \stackrel{\tau}{\rightarrow} t', k > 0 \Rightarrow p|_k|t \stackrel{\tau}{\longmapsto} p|_{k-1}|t'$.
3. *A* computation *of* $p|_k|t$ *is a maximal sequence* $p_0|_{k_0}|t_0 \stackrel{\alpha_1}{\longmapsto} p_1|_{k_1}|t_1 \stackrel{\alpha_2}{\longmapsto} p_2|_{k_2}|t_2 \stackrel{\alpha_3}{\longmapsto} \cdots$, *where* p_0, t_0, k_0 *are* p, t *and* k *respectively. Such a computation of* $p|_k|t$ *is successful if some* $p_i|_{k_i}|t_i$ *is successful and* $p_j|_{k_j}|t_j\downarrow$ *for* $j < i$.
4. $p\ emay_k\ t$ *if some computation of* $p|_k|t$ *is successful, and* $p\ emust_k\ t$ *if every computation of* $p|_k|t$ *is successful.*
5. $p \sqsubseteq_{emay} q$ *if* $\forall k, t :\ p\ emay_k\ t \Rightarrow q\ emay_k\ t$.
6. $p \sqsubseteq_{emust} q$ *if* $\forall k, t :\ p\ emust_k\ t \Rightarrow q\ emust_k\ t$.
7. $p \sqsubseteq_{etest} q$ *if* $p \sqsubseteq_{emay} q$ *and* $p \sqsubseteq_{emust} q$.

The intuition underlying this definition is best understood by contrasting it with the one given in Definition 2.2. In efficiency testing, a test is permitted to experiment with a process only for some finite amount of time chosen *a priori*. To enforce this requirement, we stipulate that the process and test share a common stopwatch (decrementing counter) that is initialized with this chosen value at the beginning of the experiment. Accordingly, each configuration is augmented with a natural number (the value of the stopwatch) representing the amount of time left before the experiment is terminated. As the experiment progresses the counter is decremented, and when the counter becomes zero the experiment is terminated; the experiment(computation) is deemed successful if a successful configuration is encountered before any under-defined configuration is encountered. The sensitivity of the resulting preorders to the amount of internal activity the processes engage in arise mainly from the second clause of Definition 3.1.2: the process engages in an internal computation, the test remains in the same state, but the counter is decremented.

We now formulate the traditional testing preorders in terms of the concepts introduced in this definition to illustrate that our definition is indeed a "reasonable" generalization. Another by-product of this formulation is the result that the $emay$ and $emust$-preorders are included in their traditional counterparts.

Proposition 3.2

1. $p \sqsubseteq_{may} q \Leftrightarrow \forall t :\ \exists k :\ p\ emay_k\ t \Rightarrow q\ emay_k\ t$

2. $p \sqsubseteq_{must} q \Leftrightarrow \forall t : \exists k : p\ emust_k\ t \Rightarrow q\ emust_k\ t$

Proof. We will only prove the first statement. The proof of the second statement is similar.

$(\Leftarrow)$: Choose and fix a test t and suppose $p\ may\ t$. We must show that $q\ may\ t$. From the fact that $p\ may\ t$ one can infer the existence of a natural number k such that $p\ emay_k\ t$. From the hypothesis it follows that $q\ emay_k\ t$ which in turn implies that $q\ may\ t$.

$(\Rightarrow)$: Suppose $p \sqsubseteq_{may} q$. Choose and fix a test t. We must either show $p\ emay_k\ t$ is true for no k or demonstrate the existence of a natural number k such that both $p\ emay_k\ t$ as well $q\ emay_k\ t$ are true. If $p\ may\ t$ is false then obviously $p\ emay_k\ t$ is true for no k. However if $p\ may\ t$ is true then it is also the case that $q\ may\ t$ as $p \sqsubseteq_{may} q$. Clearly there exists two naturals m and n such that $p\ emay_m\ t$ as well as $q\ emay_n\ t$. The maximum of m and n can be taken as the number k we are looking for.

□

We now show how the new definitions capture a notion of efficiency that may be derived from the traditional definitions. Intuitively, we may think of a process q as more efficient than a process p if given any test t, q would trigger shorter successful computations on t than p. Thus we take lengths of the successful computations (as defined in Definition 2.2) of $p||t$ to be some measure of efficiency with which a process passes a test. These ideas are formalized in the following definition.

Definition 3.3 *Let p be a process and t be a test. Further let $p||t \equiv p_0||t_0 \xrightarrow{\alpha_1} p_1||t_1 \xrightarrow{\alpha_2} p_2||t_2 \xrightarrow{\alpha_3} \cdots$ be a computation c of $p||t$ (as defined in Definition 2.2). Then the extended integer valued functions $sl, \mathcal{D}_w, \mathcal{D}_b$ are defined as follows.*

1. $sl(c) = \inf\{\ i \mid t_i \xrightarrow{w} \wedge\ (j < i \Rightarrow p_j||t_j \downarrow)\ \}$
2. $\mathcal{D}_w(p,t) = \sup\{\ sl(c) \mid c \in \mathcal{C}(p,t)\ \}$.
3. $\mathcal{D}_b(p,t) = \inf\{\ sl(c) \mid c \in \mathcal{C}(p,t)\ \}$.

Intuitively the *success length* of a computation is the *minimum* number of transitions $p||t$ must perform along this computation to reach a successful configuration without ever passing through a partially defined configuration. If this computation is unsuccessful then the success length is ∞. $\mathcal{D}_w(p,t)$ is the success length of the longest successful computation of $p||t$ and is a measure of the worst-case performance of p on t. Similarly $\mathcal{D}_b(p,t)$ is the success length of the shortest successful computation of $p||t$ and represents the best-case performance of p on t. The next proposition establishes that the $emay$ and $emust$ preorders order the processes in terms of their best-case and worst-case efficiency, respectively.

Proposition 3.4

1. $p \sqsubseteq_{emay} q \Leftrightarrow \forall t : \mathcal{D}_b(q,t) \le \mathcal{D}_b(p,t)$.
2. $p \sqsubseteq_{emust} q \Leftrightarrow \forall t : \mathcal{D}_w(q,t) \le \mathcal{D}_w(p,t)$.

Proof. We will only prove the first statement. The proof of the second is similar. Let $p \sqsubseteq_{emay} q$ and t be some test. If $\mathcal{D}_b(p,t)$ is ∞ then the required result is immediate. So suppose $\mathcal{D}_b(p,t)$ is some finite k. Then $p\ emay_k\ t$ is true. From the hypothesis, it follows that $q\ emay_k\ t$ is true as well. Then there is a computation c of $q||t$ such that $sl(c) \le k$ which implies that $\mathcal{D}_b(q,t) \le k$. This is the required result. Now suppose that for all tests t, $\mathcal{D}_b(q,t) \le \mathcal{D}_b(p,t)$. We want to establish that $p \sqsubseteq_{emay} q$. Let t, k be such that $p\ emay_k\ t$ is true. This implies that $\mathcal{D}_b(p,t) \le k$. In view of the hypothesis, we have $\mathcal{D}_b(q,t) \le k$. So it follows that $q\ emay_k\ t$ is true, which is the required result.

□

In order to facilitate reasoning about processes using these preorders, we now provide alternative characterizations that do not refer to tests. The characterizations are presented in terms of the functions defined below.

Definition 3.5 *Let p be a process and $s \in A^*$. Then $l_{min}(p,s)$ and $l_{max}(p,s)$ are the* extended-*integer-valued functions defined below.*

1. $l_{min}(p,s) = \inf\{\ |t| \mid p \xrightarrow{t} \wedge\ \hat{t} = s\ \}$.
2. $l_{max}(p,s) = \sup\{\ |t| \mid p \xrightarrow{t} \wedge\ \hat{t} = s \wedge p \Downarrow s\ \}$.

Note that when $s \notin L(p)$, $l_{min}(p,s)$ becomes ∞ and $l_{max}(p,s)$ becomes $-\infty$. Further, when $s \in L(p)$, $p \Downarrow s$ if and only if $l_{max}(p,s)$ is finite. Now we present the alternative characterizations of the $emay$-preorder and the $emust$-preorder.

Theorem 3.6

1. $p \sqsubseteq_{emay} q$ *iff* $\forall s \in A^* : l_{min}(q,s) \le l_{min}(p,s)$
2. $p \sqsubseteq_{emust} q$ *iff for every $s \in A^*$ the following holds.*

$$p \Downarrow s \Rightarrow \begin{cases} q \Downarrow s \\ q \stackrel{s}{\Rightarrow} q' \Rightarrow \exists p' : p \stackrel{s}{\Rightarrow} p' \wedge S(p') \subseteq S(q') \\ l_{max}(q,s) \le l_{max}(p,s) \end{cases}$$

Proof.

1. $(\Rightarrow)$: Let $p \sqsubseteq_{emay} q$. Choose and fix some $s \in A^*$. We would like to prove that $l_{min}(q,s) \le l_{min}(p,s)$. If $s \notin L(p)$ then $l_{min}(p,s)$ is ∞ and the required result is immediate. Now suppose that $s \in L(p)$. It is easy to see that $\mathcal{D}_b(p, s.w.nil) = l_{min}(p,s)$. Since $p \sqsubseteq_{emay} q$, by Theorem 3.6.1, it must be the case that $\mathcal{D}_b(q, s.w.nil) \le l_{min}(p,s)$ whence $l_{min}(q,s) \le l_{min}(p,s)$.

   $(\Leftarrow)$: Let $l_{min}(q,s) \le l_{min}(p,s)$ for every $s \in A^*$. Choose and fix some test t; we would like to prove that $\mathcal{D}_b(q,t) \leq \mathcal{D}_b(p,t)$. If $\mathcal{D}_b(p,t)$ is ∞ then the required inequality is immediate. Now suppose $\mathcal{D}_b(p,t)$ is some finite number k. In other words, the success length of the shortest successful computation of $p||t$ is k. Let s be the visible content of this computation. Since $l_{min}(q,s) \leq l_{min}(p,s)$ and $l_{min}(p,s)$ is finite, $l_{min}(q,s)$ is also finite and we can construct a successful computation of $q||t$ whose success length is less than or equal to k. This means that $\mathcal{D}_b(q,t) \leq k$, which is the required result.

2. ($\Rightarrow$): Let $p \sqsubseteq_{emust} q$. Choose and fix some $s \in A^*$ such that $p \Downarrow s$. Since the $emust$-preorder is contained in the $must$-preorder, the only statement that needs to be proved is $l_{max}(q,s) \leq l_{max}(p,s)$. This is trivially true if s is not a trace of q as $l_{max}(q,s) = -\infty$. So assume s is a trace of q. As p is convergent on s it follows from the fact $p \sqsubseteq_{must} q$ that s is also a trace of p. Now consider the test $E(s,n)$ for a given sequence of s of visible actions and natural number n constructed recursively as follows. $E(\epsilon, 0) = w.nil$, $E(\epsilon, n+1) = \tau.E(\epsilon, n)$, $E(at, n) = \tau.w + a.E(t,n)$. It is possible to come up with a large n such that both p and q $must$-pass $E(s,n)$ and the longest computations of both $p||E(s,n)$ and $q||E(s,n)$ have s as their visible content. If c_p and c_q are the longest computations of $p||E(s,n)$ and $q||E(s,n)$ respectively then one can infer that the success length of c_p is $l_{max}(p,s) + n$ and that of c_q is $l_{max}(q,s) + n$. As $p \sqsubseteq_{emust} q$ it must be the case that $l_{max}(p,s)+n \geq l_{max}(q,s)+n$, from which the desired result follows.

   ($\Leftarrow$): Let e be some test. We must show that $\mathcal{D}_w(q,e) \leq \mathcal{D}_w(p,e)$. If it is *not* the case that $p \; must \; e$ is true then $\mathcal{D}_w(p,e) = \infty$ and the required inequality is immediate. Now suppose that $p \; must \; e$ is true. It can be easily seen that $q \; must \; e$ is also true. So $\mathcal{D}_w(q,e)$ is finite. Let c be a computation of $q||e$ whose success length is the same as $\mathcal{D}_w(q,e)$. Further suppose that s is the visible content of c. We claim that $p \Downarrow s$. Otherwise one could construct a unsuccessful (possibly infinite) computation of $p||e$ which would contradict $p \; must \; e$. From the hypothesis, we have $l(p,s) \geq l(q,s)$. Now it is easy to construct a computation of $p||e$ whose length is more than that of c. The required result that $\mathcal{D}_w(q,e) \leq \mathcal{D}_w(p,e)$ follows immediately.

$\Box$

The $emust$-preorder also enjoys the following congruence properties.

Lemma 3.7 *Let $p \sqsubseteq_{emust} q$ and r be some process. Then the following are true.*

- $\alpha.p \sqsubseteq_{emust} \alpha.q$.
- $p|r \sqsubseteq_{emust} q|r$.
- $p[R] \sqsubseteq_{emust} q[R]$.
- $p \backslash \lambda \sqsubseteq_{emust} q \backslash \lambda$.
- $p + r \sqsubseteq_{emust} q + r$.

Note that the $emust$-preorder is already preserved under the operator $+$. This is a consequence of the fact that if $p \sqsubseteq_{emust} q$, $p \Downarrow$ and p is patient then it is also the case that q is patient. We extend $\sqsubseteq_{emay}$ and $\sqsubseteq_{emust}$ to open terms in the usual way.

Theorem 3.8 *Let t and u be terms. Then $t \sqsubseteq_{emust} u$ if and only if $t \sqsubseteq^C_{emust} u$, and $t \sqsubseteq_{emay} u$ if and only if $t \sqsubseteq^C_{emay} u$.*

The most difficult part of the proof of the above theorem involves proving these relations are preserved under recursion contexts. This was achieved by first showing that the ω-*induction* rule (see Appendix A.1) is sound with respect to these relations and then later proving that substitution under recursion is sound in "any" theory in which the ω-induction rule is sound [15]. When we compare the alternative characterizations of the $must$-precongruence and the $emust$-precongruence we notice that the only thing we additionally require is that when $p \Downarrow s$, $l_{max}(q,s) \leq l_{max}(p,s)$. Consequently, the construction of denotational models is straightforward if we follow the same approach taken by De Nicola and Hennessy [7, 15].

4. Full Abstraction

In this section we wish to prove the optimality of the $emust$-precongruence: it does not distinguish between any processes that no observer can tell apart. Such a result is typically achieved in the following manner [19]. First we decide what are the "observables of interest" for a process. Such a choice naturally induces a preorder $\preceq$ on processes: $p \preceq q$ if for all contexts $C[\,]$ the observables of $C[q]$ are "better" than the observables of $C[q]$. Second we show that $\sqsubseteq_{emust}$ and $\preceq$ are the same. Informally this means that $\sqsubseteq_{emust}$ does not make any unnecessary distinctions, and we say $\sqsubseteq_{emust}$ is *fully abstract* with respect to the observables of interest.

Since we view efficiency in terms of the amount of internal activity, a more efficient system will respond sooner than a relatively less efficient system when prompted for an interaction by the environment. Accordingly, we take "time to obtain response" to be the observable of interest for a process. Then we define a relation $\sqsubseteq_{ready}$; $\sqsubseteq^C_{ready}$ will play the role of $\preceq$ in the previous paragraph. Finally we prove that $\sqsubseteq^C_{ready}$ and $\sqsubseteq_{emust}$ are the same. The proof technique employed for establishing this result is inspired by the work of Main [11] in which he proves that the traditional $must$-preorder is the largest trace pre-congruence.

Definition 4.1 *Let p be a process and m be a natural number.*

1. *p is* responsive *if* $\exists a : p \xrightarrow{a}$.
2. *p is m-responsive if $p \Downarrow$ and ($p \xrightarrow{\tau^i} p' \Rightarrow \exists j, q$: $i + j \leq m \wedge p' \xrightarrow{\tau^j} q \wedge q$ is responsive).*
3. *p is* ready *if it is m-responsive for some m.*
4. *$p \sqsubseteq_{ready} q$ if p is m-responsive implies q is m-responsive too.*

Intuitively, a process is responsive if it is *possible* for it to respond to stimulus from its environment. Note that even if it is responsive, an unstable process is not guaranteed to respond to an external stimulus, as it can ignore that stimulus and execute the internal step. A convergent process is m-responsive if it is inevitable that it will evolve to a stable responsive state within m internal computation steps. In other words, a response to *some* external stimulus is guaranteed within m steps. Thus m is an upper bound on "time to response". Accordingly $p \sqsubseteq_{ready} q$ it implies that in the worst case, q will respond to an external stimulus "no later than" p.

Now we show that the largest precongruence contained in this relation is actually $\sqsubseteq_{emust}$.

Theorem 4.2 $\sqsubseteq^C_{ready} = \sqsubseteq_{emust}$

This theorem is proved by identifying a family of contexts that can "simulate" the behavior of all "essential" tests. If p is a process and t is an essential test then we will construct a suitable context $D_t[\,]$ such $p\ emust_k\ t$ if and only if $D_t[p]$ is k-responsive. Some of the details are formally stated below.

Definition 4.3 *If $s \in A^*, a \in A$, n is a natural number and L is a finite subset of A then the families of processes $E_1(s, a)$, $E_2(s, a, n)$ and $E_3(s, L, a)$ are defined inductively as follows.*

1. $E_1(\epsilon, a) = a.nil.$
   $E_1(bt, a) = \overline{b}.E_1(t, a) + a.nil.$
2. $E_2(\epsilon, a, n) = \tau^n.a.nil.$
   $E_2(bt, a, n) = \overline{b}.E_2(t, a, n).$
3. $E_3(\epsilon, L, a) = \sum_{\lambda \in L} \overline{\lambda}.a.nil.$
   $E_3(bt, L, a) = \overline{b}.E_3(t, L, a) + a.nil.$

Lemma 4.4 *Let $p \in \mathcal{P}$ and $s \in A^*$. Let $S = Sort(p) \cup Sort(s)$ and $a \in A \setminus (S \cup \overline{S})$. Then,*

1. *$p \Downarrow s \Leftrightarrow (p|E_1(s, a))\backslash S$ is ready.*
2. *$l_{max}(p, s) = i$ and i is finite $\Leftrightarrow \exists n_0 : \forall n \geq n_0$: $(p|E_2(s, a, n))\backslash S$ is $(n + i)$-responsive.*

Proof. We will prove only the first statement. Suppose $p \Uparrow s$. Then $(p|E_1(s, a))\backslash S \Uparrow$ and so it is not ready. Now suppose $(p|E_1(s, a))\backslash S$ is not ready. As $(p|E_1(s, a))\backslash S \overset{\epsilon}{\Rightarrow} p' \Rightarrow p' \xrightarrow{a}$, the only way the process $(p|E_1(s, a))\backslash S$ cannot be ready is if it is divergent itself, which in turn is possible only if $p \Uparrow s$.

□

In Lemma 4.4.2, one may wonder at the need for n_0. Intuitively, the necessity for this stems from the fact that while attempting to "execute" s, p may enter intermediate states from which sequences of τ's emanate that lead to states that cannot "consume" the rest of s. In order to assert that $(p|E_2(s, a, n))\backslash S$ is $(n + i)$-responsive, the length of these τ-sequences must be accounted for.

Lemma 4.5 *Let $p \in \mathcal{P}$, $s \in A^*$ such that $p \Downarrow s$ and $L \subseteq A$ be finite. Let $S = Sort(p) \cup Sort(s) \cup L$. Let $a \in A \setminus (S \cup \overline{S})$. Then $\exists p' : p \overset{s}{\Rightarrow} p' \wedge S(p') \cap L = \{\} \Leftrightarrow (p|E_3(s, L, a))\backslash S$ is not ready.*

Proof. Let $p \overset{s}{\Rightarrow} p' \wedge S(p') \cap L = \{\}$. Further let $L = \{\lambda_i\}$. As $E_3(s, L, a) \xrightarrow{s} E_3(\epsilon, L, a)$, $(p|E_3(s, L, a))\backslash S \overset{\epsilon}{\Rightarrow} (p'|\sum \overline{\lambda}_i.a.nil)\backslash S$, which is a stable, unresponsive process since S contains L as well as the sort of the process p. Consequently $(p|E_3(s, L, a))\backslash S$ is not ready. Now suppose $(p|E_3(s, L, a))\backslash S$ is not ready. Using the fact that $p \Downarrow s$, it is not difficult to see that $\exists p' : p \overset{s}{\Rightarrow} p' \wedge S(p') \cap L = \{\}$.

□

Lemma 4.6 $p \sqsubseteq^C_{ready} q \Rightarrow p \sqsubseteq_{emust} q.$

Proof. Let $p \sqsubseteq^C_{ready} q$. We will use Theorem 3.6 to prove that $p \sqsubseteq_{emust} q$. Let $S_p, S_q, S_{p,q}$ stand for the sorts of the process p, q and $S_p \cup S_q$ respectively. Let s be a sequence of visible actions such that $p \Downarrow s$. Now we will prove $q \Downarrow s$. Let S be a finite set that includes $S_{p,q}$ and all the actions that occur in s; let a be some visible action not in $S \cup \overline{S}$.

By Lemma 4.4.1, $(p|E_1(s, a))\backslash S$ is ready . As $p \sqsubseteq^C_{ready} q$, it follows that

$$(p|E_1(s, a))\backslash S \sqsubseteq_{ready} (q|E_1(s, a))\backslash S$$

. Thus $(q|E_1(s, a))\backslash S$ is ready as well. By applying Lemma 4.4.1 once again, we obtain $q \Downarrow s$, as desired.

Assume $p \Downarrow s$. From the argument made in the previous paragraph we have $q \Downarrow s$. Now we would like to establish $l_{max}(p, s) \geq l_{max}(q, s)$. If $l_{max}(q, s) = -\infty$, then the required result is immediate. Assume $l_{max}(q, s) = i$ for some finite i. By Lemma 4.4.2, $(q|E_2(s, a, n))\backslash S$ is $(n + i)$-responsive for some suitably large n. As $p \Downarrow s$, we have $(p|E_2(s, a, n))\backslash S$ is ready. Since $(p|E_2(s, a, n))\backslash S \sqsubseteq_{ready} (q|E_2(s, a, n))\backslash S$ we can infer that $(p|E_2(s, a, n))\backslash S$ is $n + j$-responsive for some $j \geq i$. By using Lemma 4.4.2 once again we get the required result that $l_{max}(p, s) \geq l_{max}(q, s)$.

Now let there exist some q' such that $q \stackrel{s}{\Rightarrow} q'$. Let L stand for the set $S \backslash S(q')$. Obviously $L \cap S(q') = \{\}$. By Lemma 4.5, $(q|E_3(s, L, a)) \backslash S$ is not ready. So $(p|E_3(s, L, a)) \backslash S$ is not ready as well. By applying Lemma 4.5 once again we infer the existence of a p' such that $p \stackrel{s}{\Rightarrow} p'$ and $S(p') \cap L = \{\}$. But L is $S \backslash S(q')$. Thus it follows that $S(p') \subseteq S(q')$.

In view of Theorem 3.6, we have now established $p \sqsubseteq_{emust} q$.

□

Lemma 4.7 $p \sqsubseteq_{emust} q \Rightarrow p \sqsubseteq^{C}_{ready} q$.

Proof. We will prove $\sqsubseteq_{emust}$ is contained in $\sqsubseteq_{ready}$; the required result would then follow immediately. Suppose $p \sqsubseteq_{emust} q$ and p is ready. This implies that $p \Downarrow$ which in turn implies that $q \Downarrow$ as well. If q is not ready, it could be only due to a presence of q' such that $q \stackrel{\epsilon}{\Rightarrow} q'$ and $q' \stackrel{\tau}{\not\rightarrow}$. As $p \sqsubseteq_{emust} q$, there must be a p' such that $p \stackrel{\epsilon}{\Rightarrow} p'$ and $p' \stackrel{\tau}{\not\rightarrow}$. This would contradict the assumption that p is ready. Thus it must be the case that q is ready. From the fact that $l_{max}(q, \epsilon) \leq l_{max}(p, \epsilon)$ the required result follows immediately.

□

It is also possible to establish an optimality result of the $emay$ preorder for totally convergent processes with respect to "best-case" response.

5. Other Efficiency Preorders

In this section we compare the existing literature on bisimulation-based efficiency preorders with the (testing-based) efficiency preorders we have defined in this paper. It should be noted that the existing efficiency-based preorders are defined over the same language CCS defined in Section 2 but without the process Ω.

The following definition was proposed in [1]. It relates processes on the basis of their best-case performance as well as worst-case performance.

Definition 5.1 *$R \subseteq \mathcal{P} \times \mathcal{P}$ is called an* efficiency prebisimulation *if for every $\langle p, q \rangle \in R$ the following conditions hold.*

$$\begin{array}{l} p \stackrel{a}{\rightarrow} p' \Rightarrow \exists q' : \; q \stackrel{a}{\rightarrow} q' \wedge p' R q' \\ p \stackrel{\tau}{\rightarrow} p' \Rightarrow (\exists q' : \; q \stackrel{\tau}{\rightarrow} q' \wedge p' R q') \vee p' R q \\ q \stackrel{\alpha}{\rightarrow} q' \Rightarrow \exists p' : \; p \stackrel{\hat{\alpha}}{\Rightarrow} p' \wedge p' R q' \end{array}$$

Let $\leq_1$ stand for the largest efficiency prebisimulation. With the view towards arriving at a relation which is coarser than $\leq_1$, the following definition was introduced in [2].

Definition 5.2 *$R \subseteq \mathcal{P} \times \mathcal{P}$ is called an* elaboration *if for every $\langle p, q \rangle \in R$ the following conditions hold.*

$$\begin{array}{l} p \stackrel{\alpha}{\rightarrow} q' \Rightarrow \exists q' : \; q \stackrel{\hat{\alpha}}{\Rightarrow} q' \wedge p' R q' \\ q \stackrel{\alpha}{\rightarrow} q' \Rightarrow \exists p' : \; p \stackrel{\alpha}{\Rightarrow} p' \wedge p' R q' \end{array}$$

$$\begin{array}{llcl}
A1 & x + x & = & x \\
A2 & x + y & = & y + x \\
A3 & x + (y + z) & = & (x + y) + z \\
A4 & x + nil & = & x \\
A5 & \multicolumn{3}{l}{\text{Let } p, q \text{ denote } \sum \alpha_i.p_i \; \{+\Omega\}, \sum \beta_j.q_j \; \{+\Omega\} \text{ respectively. Then}} \\
\multicolumn{4}{l}{p|q = \sum \alpha_i.(p_i|q) + \sum \beta_j.(p|q_j) + \sum_{\alpha_i = \overline{\beta_j} \in A} \tau.(p_i|q_j)} \\
\multicolumn{4}{c}{+\{\; \Omega \mid \Omega \text{ is a summand of } p \text{ or } q \;\}} \\
A6 & nil \backslash \lambda & = & nil \\
A7 & (\alpha.x) \backslash \lambda & = & \begin{cases} nil & \alpha \in \{\lambda, \overline{\lambda}\} \\ \alpha.(x \backslash \lambda) & \text{otherwise} \end{cases} \\
A8 & (x + y) \backslash \lambda & = & (x \backslash \lambda) + (y \backslash \lambda) \\
A9 & nil[S] & = & nil \\
A10 & (\alpha.x)[S] & = & S(\alpha).(x[S]) \\
A11 & (x + y)[S] & = & x[S] + y[S] \\
A12 & \Omega & \sqsubseteq & x \\
A13 & \Omega \backslash \lambda & \sqsubseteq & \Omega \\
A14 & \Omega[S] & \sqsubseteq & \Omega \\
A15 & \tau.x & \sqsubseteq & x \\
A16 & t[fix(x : t)/x] & \sqsubseteq & fix(x : t) \\
 & & & \\
B1 & \tau.x + \tau.y & \sqsubseteq & x \\
B2 & \alpha(\tau.x + \tau.y) & \sqsubseteq & \alpha.x + \alpha.y \\
B3 & \tau(\alpha.x + \alpha.y + z) & \sqsubseteq & \alpha.x + \tau(\alpha.y + z)
\end{array}$$

Figure 2. The Proof System E$_1$

$$\begin{array}{llcl}
C1 & \alpha.x + \alpha.y & = & \alpha(\tau.x + \tau.y) \\
C2 & x + \tau.y & \sqsubseteq & \tau(x + y)
\end{array}$$

Figure 3. The Proof System E$_2$

Let $\leq_2$ stand for the largest elaboration. It compares processes only on the basis of their worst-case performance. It was also shown in [2] that $\leq_1$ is strictly included in $\leq_2$. Now we state a proposition which establishes the connection between the preorders stated above and the testing-based efficiency preorders introduced in the earlier section.

Proposition 5.3 *$\leq_1$ and $\leq_2$ are strictly included in $\sqsubseteq_{etest}$ and $\sqsubseteq_{emust}$ respectively.*

Having seen that $\leq_1$ and $\leq_2$ are analogues of $\sqsubseteq_{etest}$ and $\sqsubseteq_{emust}$ respectively one might wonder which relation is the analogue of $\sqsubseteq_{emay}$. An answer to this question and how a coherent picture of bisimulation-based efficiency preorders can be obtained is given in Appendix A.2

6. Proof Systems

In this section we provide a sound and complete axiomatization of the precongruences $\sqsubseteq_{emay}$ and $\sqsubseteq_{emust}$ over

$D1$	x	$\sqsubseteq$	$x + y$
$D2$	$\alpha(x + y)$	$=$	$\alpha.x + \alpha.y$
$D3$	$a.x + a.\tau.y$	$=$	$a.x + \tau.a.y$
$F1$	$x + y$	$\sqsubseteq$	x
$F2$	$\alpha(x + y)$	$=$	$\alpha.x + \alpha.y$
$F3$	$\tau.a.x + \tau.y$	$=$	$a.\tau.x + \tau.y$

Figure 4. The additional axioms E_3 and E_4

arbitrary processes: two closed terms are related by these precongruences if and only if this can be derived *syntactically* from the axioms using the laws of inequational reasoning (Appendix A.1).

Let E_1, E_2 denote the set of axioms given in Figures 2, 3 respectively. Let E_3 and E_4 stand for the set of axioms $D1$ through $D3$ and $F1$ through $F3$ respectively. Further let $E_{i,j}$ stand for set of axioms $E_i \cup E_j$. It has been established in [15] that $E_{1,2}$ is sound and complete for the traditional $must$ precongruence. Our aim, however, is to establish the following algebraic characterizations of the testing-based efficiency precongruences.

Theorem 6.1 *Let p, q be processes. Then the following hold.*

1. $p \precsim_{emay} q \Leftrightarrow \vdash_{E_{1,3}} p \sqsubseteq q$.
2. $(\forall s \in A^* : p \Downarrow s \Rightarrow (q \Downarrow s \wedge l_{max}(q, s) \leq l_{max}(p, s))) \Leftrightarrow \vdash_{E_{1,4}} p \sqsubseteq q$
3. $p \precsim_{emust} q \Leftrightarrow \vdash_{E_{1,2}} p \sqsubseteq q$ *and* $\vdash_{E_{1,4}} p \sqsubseteq q$.

The proof of this theorem is quite involved; however, it closely follows the similar proof from [15]. The soundness of the infinitary ω-induction rule follows from the following facts. If $p \in \mathcal{P}$ and $s \in A^*$ then there exists a "finite approximation" d of p such that $l_{min}(d, s) = l_{min}(p, s)$. Similarly if $p \Downarrow s$ then one can find a finite approximation d of p such that $d \Downarrow s$ and $l_{max}(d, s) = l_{max}(p, s)$. The proof of completeness, as usual, involves isolating certain terms as *normal forms* and showing how any arbitrary term can be rewritten as a normal form using the axioms, and is somewhat complex. For simplicity we will restrict our attention to *finite*, *totally convergent* processes.

Definition 6.2

1. *nil is a* sum form. $\Sigma\alpha_i.p_i$ *is a* sum form *if each p_i is also a sum form.*
2. *nil is a l_{min}-normal form. $\Sigma\tau^{i_a}.a.p_a$ is a l_{min}-normal form if each p_a is.*
3. *nil is a l_{max}-normal form. $\tau^i + \Sigma a.p_a$ is a l_{max}-normal form if each p_a is.*

Suppose we are given two finite, totally convergent terms p and q such that $p \precsim_{emay} q$. We now explain how we can establish that $\vdash_{E_{1,3}} p \sqsubseteq q$. By usual techniques [12], we can find sum forms p_s, q_s such that $\vdash_{E_{1,3}} p = p_s$ and $\vdash_{E_{1,3}} q = q_s$. In order to transform p to p_s and q to q_s we only use axioms $A1$ through $A11$. We can now transform p_s and q_s to l_{min}-normal forms p_n and q_n respectively using mainly the axioms $D2$,$D3$ and $A1 - A3$. Finally, it can be shown that $\vdash_{E_{1,3}} p_n \sqsubseteq q_n$ with the aid of the axioms $A15, D1$ and $D2$. By a similar argument, we can show $\vdash_{E_{1,3}} p \sqsubseteq q$ if we are given two finite, totally convergent terms p and q such that for all $s \in A^*$,

$$p \Downarrow s \Rightarrow (q \Downarrow s \wedge l_{max}(q, s) \leq l_{max}(p, s))$$

A noteworthy feature of the Theorem 6.1.3 is that it permits the separation of reasoning about functionality from reasoning about efficiency.

We end this section with a proposition which presents an axiom that can be derived from both of the proof systems $E_{1,3}$ and $E_{1,4}$; it will be used to reason about the example presented in the Section 7.

Proposition 6.3 $\tau.a.x \sqsubseteq a.\tau.x$ *is a derived axiom under both $E_{1,3}$ and $E_{1,4}$.*

Proof. In this proof, to conserve space, we will write "$x \sqsubseteq (Ax)\ y$ to mean x can be rewritten as y with the axiom labeled "Ax" playing the key role in the transformation. We begin with the proof of $\tau.a.x \sqsubseteq a.\tau.x$ under $E_{1,3}$. $\tau.a.x \sqsubseteq (D1)\tau.a.x + a.\tau.nil = (D3)a.\tau.nil + a.\tau.x = (D2)a.\tau.(x + nil) = (A4)a.\tau.x$ Now we will establish $\tau.a.x \sqsubseteq a.\tau.x$, using the proof system $E_{1,4}$. $\tau.a.x = (A4)\tau.a.(x + nil) = (F2)\tau.a.x + \tau.a.nil = (F3)a.\tau.x + \tau.a.nil \sqsubseteq (F2)a.\tau.x$

□

7. An Example

This section is devoted to illustrating the use of efficiency preorders to reason formally about the specification of a system and its putative implementations. The system we consider is that of a *dictionary*; given a *key* it might either give its *meaning* or reply that such a word is not in its database. An abstract *functional* specification of the system is given by the process *Spec* defined below.

$$Spec \Leftarrow key.\overline{meaning}.Spec + key.\overline{no}.Spec$$

Henceforth we will use k, m, n for the actions *key*, *meaning* and *no*. Now we give two "implementations" of the above specification by means of two processes *Imp1* and *Imp2*. Both the implementations consist of two components: a *dictionary manager* and the *database*. In the first implementation the manager receives the input key through the channel *k* and passes it along to the database through the channel *ik*. The database then initiates a search for the word and nondeterministically either returns the meaning of the word to the manager through the channel *im* or signals the absence of the key to the manager through the channel *in*. On receipt of this signal the dictionary manager informs the environment of the answer to the query.

$$\textit{Imp1} \Leftarrow (DM1|DB1) \setminus \{\textit{ik}, \textit{im}, \textit{in}\}$$

$$\textit{DM1} \Leftarrow k.\overline{\textit{ik}}.(\textit{im}.\overline{m}.DM1 + \textit{in}.\overline{n}.DM1)$$

$$\textit{DB1} \Leftarrow \textit{ik}.(\tau.\overline{\textit{im}}.DB1 + \tau.\overline{\textit{in}}.DB1)$$

In the second implementation, *Imp2*, the database conveys the answer of the query it received from the manager directly to the environment. The database also sends an acknowledgment along *reset* to the manager to indicate that the answer to the query has been conveyed to the environment; this ensures that the system does not receive a query before the earlier one has been answered.

$$\textit{Imp2} \Leftarrow (DM2|DB2) \setminus \{\textit{ik}, \textit{reset}\}$$

$$\textit{DM2} \Leftarrow k.\overline{\textit{ik}}.\textit{reset}.DM2$$

$$\textit{DB2} \Leftarrow \overline{\textit{ik}}.(\tau.\overline{m}.\overline{\textit{reset}} + \tau.\overline{n}.\overline{\textit{reset}})$$

It can be shown that *Spec*, *Imp1* and *Imp2* are *testing*-equivalent in the traditional sense with respect to each other. This establishes the functional correctness of the implementations with respect to the specification. Now we will show that *Imp2* is strictly more efficient than *Imp1* both in the worst case as well as best-case performance by establishing *Imp1* $\sqsubseteq\!\!\!\sim_{emay}$ *Imp2* and *Imp1* $\sqsubseteq\!\!\!\sim_{emust}$ *Imp2*. To accomplish this goal we will use the proof theory developed in the Section 6.

Using the proof system E_1 we can prove

$$\textit{Imp1} = k.\tau.(\tau.\tau.\overline{m}.\textit{Imp1} + \tau.\tau.\overline{n}.\textit{Imp1})$$

$$\textit{Imp2} = k.\tau.(\tau.\overline{m}.\tau.\textit{Imp2} + \tau.\overline{n}.\tau.\textit{Imp2})$$

By subsequent use of *F2* (which is same as *D2*), the derived axiom mentioned in Proposition 6.3 and the law of substitutivity we can show *Imp1* $\sqsubseteq$ *Imp2*. Note that all the axioms and rules of inference used in this exercise belong to both $E_{1,3}$ and $E_{1,4}$. From this we can infer that *Imp1* $\sqsubseteq\!\!\!\sim_{emay}$ *Imp2* and *Imp1* $\sqsubseteq\!\!\!\sim_{emust}$ *Imp2*. Furthermore, by examining $l_{max}(\textit{Imp1}, k)(= 4)$ and $l_{max}(\textit{Imp2}, k)(= 3)$ we infer that *Imp2* $\not\sqsubseteq\!\!\!\sim_{emust}$ *Imp1*. Similarly, by examining $l_{min}(\textit{Imp1}, k\overline{m})(= 5)$ and $l_{min}(\textit{Imp2}, k\overline{m})(= 4)$ we infer that *Imp2* $\not\sqsubseteq\!\!\!\sim_{emay}$ *Imp1*.

Observe that *Spec* is not $\leq_i$-related to either of *Imp1* and *Imp2* for any $i = 1, 2, 3$. Even though *Imp1* and *Imp2* are observationally equivalent [12] they are not related by any of the existing $\leq_i$ preorders. Thus this example demonstrates that these bisimulation-based efficiency preorders in some sense make too few identifications.

8. Conclusions

In this paper we developed an algebraic theory of efficiency preorders grounded on the testing framework of De Nicola and Hennessy [15]. If a process p is less than q under this theory, it means that q is more efficient and more deterministic than p. We formulated this theory by modifying the traditional testing framework so that tests become sensitive to the precise number of internal steps processes engage in. We observed that the resulting “may” and “must” preorders relate the processes on the basis of their best-case efficiency and worst-case efficiency respectively. We then identified the largest precongruences contained in these preorders and provided alternative characterizations which do not refer to tests. This necessarily involved justifying the infinitary ω-induction rule. Later we demonstrated the optimality of these precongruences by proving that they are the largest “reasonable” precongruences respecting efficiency. The optimality was further reaffirmed by the fact they subsume the existing bisimulation-based efficiency preorders [1, 2]. Finally we illustrated the use of these precongruences and the proof theory by carrying out a very simple exercise in system verification. This example also served as a vehicle for demonstrating a potential shortcoming of existing bisimulation-based efficiency preorders.

Other researchers have investigated notions of “speed” in setting of real-time models of concurrency in which time passage is explicitly captured in special actions. Our theory may be altered in a straightforward way to model notions of “faster than” for these models; when we did this for TPL [9], a CCS-like process algebra with discrete time and the *maximal progress* assumption, we found that the $emust$-preorder coincides with the $must$-preorder as defined by Hennessy and Regan. This surprising result merits further study and we intend to pursue it in the future [14]. In a non-interleaving semantic framework, testing-based “faster than” relations have been developed by Vogler [16, 17]. The idea of having tests with time-bounds appears in [16], while interesting consequences of permitting simultaneous execution of actions on the determination of when a system works faster than another are explored in [17]. But none of them contain an algebraic theory or full abstractness result as we have developed here. Theories for relating processes on the basis of their speed also appear in [13, 5]. None of them present any kind of full abstractness result.

Acknowledgements: We would like to thank S. Arun-Kumar for several insightful comments.

References

[1] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29(8):737–760, 1992.

[2] S. Arun-Kumar and V. Natarajan. Conformance: A precongruence close to bisimilarity. In *International Workshop*

on Structures in Concurrency Theory, Springer Workshop in Computing Series, May 1995.

[3] J. Baeten and W. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, England, 1990.

[4] S. Brookes, C. Hoare, and A. Roscoe. A theory of communicating sequential processes. *Journal of the Association for Computing Machinery*, 31(3):560–599, July 1984.

[5] R. Cleaveland and A. Zwarico. A theory of testing for real time. In *Sixth Annual Symposium on Logic in Computer Science (LICS '91)*, pages 110–119, Amsterdam, July 1991. IEEE Computer Society Press.

[6] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 1987.

[7] M. Hennessy. Acceptance trees. *JACM*, 32(4):896–928, October 1985.

[8] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, Boston, 1988.

[9] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, (117):221–239, 1995.

[10] C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, London, 1985.

[11] M. Main. Trace, failure and testing equivalences for communicating processes. *International Journal of Parallel Programming*, 1987.

[12] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[13] F. Moller and C. Tofts. Relating processes with respect to speed. In J. Baeten and J. Groote, editors, *CONCUR '91*, volume 527 of *Lecture Notes in Computer Science*, pages 424–438, Amsterdam, Aug. 1991. Springer-Verlag.

[14] V. Natarajan and R. Cleaveland. Predictability of real-time systems: A process algebraic approach. Copy of the manuscript can be obtained by contacting the authors., 1996.

[15] R. D. Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, (34):83–133, 1984.

[16] W. Vogler. Timed testing of concurrent systems. In *ICALP '93*, volume 700 of *LNCS*. Springer-Verlag, 1993.

[17] W. Vogler. Faster asynchronous systems. In *CONCUR '95*, volume 962 of *LNCS*, pages 299–313. Springer-Verlag, 1995.

[18] D. Walker. Bisimulations and divergence in CCS. *Information and Computation*, 85(2):202–241, 1990.

[19] G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993.

A. Appendix

A.1 Laws of inequational reasoning

The following are the laws of inequational reasoning [15].

1. (Equality) $\frac{t=u}{t\sqsubseteq u,\, u\sqsubseteq t}$ $\quad \frac{t\sqsubseteq u,\, u\sqsubseteq t}{t=u}$
2. (Partial Order) $\frac{t\sqsubseteq u,\, u\sqsubseteq r}{t\sqsubseteq r}$ $\quad \frac{}{t\sqsubseteq t}$
3. (Substitutivity)

$$\frac{t\sqsubseteq u}{t\rho\sqsubseteq u\rho}$$

$$\frac{t\sqsubseteq u}{\mathit{fix}(x:t)\sqsubseteq \mathit{fix}(x:u)}$$

$$\frac{t_i\sqsubseteq u_i, 1<i<k}{op(t_1,\cdots,t_k)\sqsubseteq op(u_1,\cdots,u_k)}$$

4. (ω-induction) $\frac{\forall d\in Fin(t)\colon\, d\sqsubseteq u}{t\sqsubseteq u}$

If $\prec$ is the least pre-congruence generated from $A12$ and $A16$ Figure.2 then $Fin(t)$ is the set of all finite terms d such that $d \prec t$.

$G1$	$\alpha(x+\tau.x)$	$\sqsubseteq$	$\alpha.x$
$G2$	$\tau.x$	$\sqsubseteq$	$\tau.x+x$
$G3$	$\alpha.(x+\tau.y)$	$\sqsubseteq$	$\alpha(x+\tau.y)+\alpha.y$
$G4$	$x\sqsubseteq x+y+z$	$\Rightarrow$	$x\sqsubseteq x+z$
$G5$	$\tau.x+x$	$\sqsubseteq$	$\tau.x$
$G6$	$\alpha(x+\tau.y)+\alpha.y$	$\sqsubseteq$	$\alpha(x+\tau.y)$
$G7$	$\alpha.x$	$\sqsubseteq$	$\alpha(x+\tau.x)$
$H1$	$\alpha.\tau.x$	$\sqsubseteq$	$\alpha.x$
$H2$	$\tau.x+x$	$\sqsubseteq$	$\tau.x$
$H3$	$\alpha(x+\tau.y)+\alpha.y$	$\sqsubseteq$	$\alpha(x+\tau.y)$

Figure 5. The Proof System for $\leq$-precongruences

A.2 Bisimulation-based efficiency preorders

We begin with the definition of the analogue of $\sqsubseteq_{emay}$.

Definition A.1 *$R \subseteq \mathcal{P}\times\mathcal{P}$ is called an* strong elaboration *if for every $\langle p,q\rangle \in R$ the following conditions hold.*

$$\begin{array}{l} p \xrightarrow{a} p' \Rightarrow \exists q' :\ q \xrightarrow{a} q' \wedge p'Rq' \\ p \xrightarrow{\tau} p' \Rightarrow (\exists q' :\ q \xrightarrow{\tau} q' \wedge p'Rq') \vee p'Rq \\ q \xrightarrow{\alpha} q' \Rightarrow \exists p' :\ p \overset{\hat{\alpha}}{\Rightarrow} p' \wedge p'Rq' \end{array}$$

Let $\leq_3$ stand for the largest strong elaboration. It is easy to see that $\leq_1 \subset \leq_3$ and $\leq_3$ and $\leq_2$ incomparable. Moreover $\leq_3$ is preserved under all operators except for $+$ and $\leq_3^C$ can be obtained in the usual manner. Now we turn our attention to the algebraic characterization of these relations over finite processes. It has been shown [1] the set of axioms $A1$ through $A11$ of Figure 2 along with $G1$ through $G4$ of Figure 5 characterize $\leq_1$ over finite processes. Similary [2] has established that the axioms $A1$ through $A11$ of Figure 2 along with $G2, G3, H1, H2, H3$ of Figure 5 characterize $\leq_2$.

Now we state the main theorem of this section which presents the proof system for $\leq_3$ and a slightly modified proof system for $\leq_2$. The advantage of this formulation is that it emphasizes the similarity between them and demonstrates how they can be obtained by smoothly extending the proof system for $\leq_1$. The first point of the theorem is taken from [1]

Theorem A.2 *Let p, q be finite processes. Then*

1. $A1-A11, G1-G4 \vdash p \sqsubseteq q \Leftrightarrow p \leq_1^C q.$
2. $A1-A11, G1-G6 \vdash p \sqsubseteq q \Leftrightarrow p \leq_2^C q.$
3. $A1-A11, G1-G4, G7 \vdash p \sqsubseteq q \Leftrightarrow p \leq_3^C q.$