LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Relating Processes with Respect to Speed

by

Faron Moller
Chris Tofts

Relating Processes with Respect to Speed

LFCS Report Series **ECS-LFCS-91-143**

LFCS **March 1991**

Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Relating Processes With Respect To Speed

Faron Moller

Chris Tofts

Department of Computer Science
University of Edinburgh

Abstract

In this paper, we consider the problem of defining a preorder on concurrent processes which will distinguish between functionally behaviourally equivalent processes which operate at different speeds. As our basic framework, we use a subset of the calculus TCCS of [Mol90], a language for describing concurrent processes involving timing constraints.

There is an anomaly in timed process calculi such as TCCS which nullifies the possibility of defining such a preorder which is a precongruence. This anomaly arises due to the nature of the constructs in the calculus which force events to be executed without delay. To rectify this conflict, we define and motivate the above mentioned subcalculus, which we call ℓTCCS (*loose* TCCS), and define our relation over this language. ℓTCCS is precisely TCCS where all events may delay indefinitely before executing. We demonstrate why this is necessary in order for any sensible *faster than* relation to be a precongruence.

Upon providing the semantic definition of our *"faster than"* relation, we give results on the precongruicity of the relation and present a set of inequational laws.

1 Introduction

In [Mol90], the authors introduced a language for the expression and analysis of timing constraints within concurrent processes. The language TCCS, the *Temporal Calculus of Communicating Systems*, was founded on Milner's *Calculus of Communicating Systems* CCS of [Mil80, Mil89]. The syntax and operational semantics of the calculus were introduced, and an observational congruence was defined based on Park's notion of bisimulation from [Par81]. Equational laws were provided for reasoning about the defined semantic congruence, and several examples were provided to illustrate the utility of the calculus in reasoning about concurrent processes involving timing constraints.

A missing feature in this work was a study on the relative speeds of functionally behaviourally equivalent processes. Two processes were deemed equivalent roughly when they were identical in both their functional *and* their temporal behaviour, as viewed by an external observer. The purpose of the present paper is to fill in this missing aspect by defining a *"faster than"* precongruence which will hold between two process terms if they are functionally behaviourally equivalent, but that the first term can execute its function faster (*ie*, sooner) than the second term.

The reason that the study of a *faster than* precongruence was not addressed in TCCS was that, as is explained in the next section of this paper, the expressiveness of TCCS is too powerful to admit such a notion. In order to define a sensible notion of relative speeds, we need to restrict ourselves to a subset of TCCS.

In the next section of this paper, we outline the theory of TCCS, and explain the problem with discussing speed in the full calculus. We use this discussion to motivate and define our subcalculus with which we shall be working. Next, we formally define our semantic precongruence, and relate it to the equivalence defined in [Mol90] for TCCS. We then go on to present some sound inequational laws for reasoning algebraically about processes and their relative speeds.

2 The Calculus TCCS

In this section, we give a brief introduction to the calculus TCCS by presenting the syntax and transitional semantics of the language, along with a behavioural equivalence defined using the semantic definition of the calculus. The description is self contained; however, the reader is urged to refer to [Mol90] for the full treatment of the language. We then present the difficulty which arises in trying to define a sensible *faster than* relation, and motivate a subcalculus of TCCS to be used for this purpose.

2.1 Syntax

The language TCCS is a timed extension of CCS. To define the syntax of the language, we first presuppose a set Λ of atomic action symbols not containing τ, and we define $\mathrm{Act} = \Lambda \cup \{\tau\}$. We assume that Λ can be partitioned into two equinumerous sets with a "complementation" bijection $\bar{\cdot}$ between them. These complementary actions form the basis of the communication method in our calculus, analogous to CCS. We also take $\mathcal{T} = \{1, 2, 3, \ldots\}$ to represent divisions in time, and presuppose some set Var of process variables.

The collection of TCCS expressions, ranged over by P, is then defined by the following BNF expression where we take $a \in \mathrm{Act}$, $L \subseteq \Lambda$, $X \in \mathrm{Var}$, $t \in \mathcal{T}$, S ranging over *relabelling functions*, those $S : \Lambda \to \Lambda$ such that $\overline{S(a)} = S(\bar{a})$ and $S(\tau) = \tau$, and $\tilde{x}$, $\tilde{P}$ representing (equal-length) vectors of process variables $(x_1, x_2, \ldots, x_n)$ and expressions $(P_1, P_2, \ldots, P_n)$ respectively.

$$\begin{aligned} P \;::=\; & \mathbf{0} \mid X \mid a.P \mid (t).P \mid \delta.P \\ & \mid P \oplus P \mid P + P \mid P \mid P \mid P \backslash L \mid P[S] \mid \mu_i \tilde{x}.\tilde{P} \end{aligned}$$

The informal interpretation of these is as follows.

- $\mathbf{0}$ represents the nil process, which can neither proceed with any action, nor proceed through time.
- X represents the process bound to the variable X.
- $a.P$ represents the process which can perform the action a and evolve into the process P upon so doing.
- $(t).P$ represents the process which will evolve into the process P after exactly t units of time.
- $\delta.P$ represents the process which behaves as the process P, but is willing to wait any amount of time before actually proceeding. The understanding of this process is in fact that it is wanting to synchronise or communicate with its environment, but is willing to wait until such time as the environment is ready to participate in such a communication.
- $P + Q$ represents a choice between the two processes P and Q. The process will behave as the process P or the process Q, with the choice being made at the time of the first action. Thus for instance any initial passage of time must be allowed by both P and Q. We shall refer to this operator as *strong* choice.
- $P \oplus Q$ represents a slightly different notion of choice between the two processes P and Q. The process will behave as the process P or the process Q, with the choice being made at the time of

the first action, or else at the occurrence of a passage of time when only one of the operands may allow the time passage to occur. In this case, the second *"stopped"* process will be dropped from the computation. We shall refer to this operator as *weak* choice.

- $P \mid Q$ represents the parallel composition of the two processes P and Q. Each of the processes may do any actions independently, or they may synchronise on complementary actions, resulting in a τ action. Any passage of time must be allowed and recorded by each of P and Q.

- $P\backslash L$ represents the process P with the actions named in L restricted away, that is, not allowed to occur.

- $P[S]$ represents the process P with its actions relabelled by the relabelling function S.

- $\mu_i\widetilde{x}.\widetilde{P}$ represents the solution x_i taken from the solutions to the mutually recursive definitions of the processes $\widetilde{x}$ defined as particular solutions to the equations $\widetilde{x} = \widetilde{P}$.

Some points worth noting which arise from the above informal description are as follows.

- The process $\mathbf{0}$ acts as a deadlock process in that it cannot perform any actions, nor witness any passage of time. Hence, by the descriptions above of the strong and weak choice operators $+$ and $\oplus$, and the parallel composition operator $|$, the constant $\mathbf{0}$ acts as an annihilator with respect to adding or composing with time-guarded processes, and as a unit with respect to the weak choice operator $\oplus$. Thus in particular, local temporal deadlock will imply global deadlock — if only time derivations are possible from each component of a parallel composition involving $\mathbf{0}$, then the whole composite process is deadlocked. Hence, of interest is the derived *nontemporal deadlock* process $\delta.\mathbf{0}$, which will allow any time to pass, but can never perform any actions. This process thus stands as a unit with respect to the strong choice operator $+$ and the parallel composition operator $|$. We shall henceforth abbreviate this process to $\underline{0}$.

- The description given above of the delay prefix δ is such that it is only meaningful to follow it with action terms. $\delta.P$ represents a process which is delaying the *actions* of P until the environment in which the process is executing will allow the actions to proceed. Thus for instance, the process $\delta.(1).a.\mathbf{0}$ can never perform its action a, as it can never get past the delaying δ. Hence, this process will be identified with $\underline{0}$, the nontemporal deadlock. Of importance then is the delayed action prefix $\delta.a.P$, which we henceforth abbreviate to $\underline{a}.P$.

We shall occasionally omit the dot symbol when applying the prefix operators, and also drop trailing $\mathbf{0}$'s, thus for instance rendering $\delta.a.\mathbf{0} + (t).b.\mathbf{0}$ as $\delta a + (t)b$ (or as $\underline{a} + (t)b$). Finally, we shall allow

ourselves to specify processes definitionally, by providing recursive definitions of processes. For example, we shall write $A \stackrel{\text{def}}{=} a.A$ rather than $A \stackrel{\text{def}}{=} \mu x.a.x$.

2.2 Semantics

The semantics of TCCS is *transition based*, structurally presented in the style of [Plo81], outlining what actions and time delays a process can witness. In order to define our semantics however, we must first define a syntactic predicate which will allow us to specify when a process must stop delaying its computation within a particular amount of time. This is done using the following function on TCCS terms.

Definition 2.1 *The function stop* : TCCS $\rightarrow \{0, 1, 2, \ldots, \omega\}$ *defines the maximum delay which a process may allow before forcing a computation (or deadlock) to occur. Formally, this function is defined as follows:*

$$
\begin{array}{ll}
stop(\dot{0}) = 0 & stop(P \oplus Q) = \max(stop(P), stop(Q)) \\
stop(X) = 0 & stop(P + Q) = \min(stop(P), stop(Q)) \\
stop(a.P) = 0 & stop(P \mid Q) = \min(stop(P), stop(Q)) \\
stop((s).P) = s + stop(P) & stop(P \backslash L) = stop(P) \\
stop(\delta.P) = \omega & stop(P[S]) = stop(P) \\
 & stop(\mu_i \tilde{x}.\tilde{P}) = stop(P_i\{^{\mu \tilde{x}.\tilde{P}}/_{\tilde{x}}\})
\end{array}
$$

This definition is well defined as long as all recursive variables are guarded by action or time prefixes.

In Figure 1, we present the operational rules for our language. They are presented in a natural deduction style, and are to be read as follows: if the transition(s) above the inference line can be inferred, then we can infer the transition below the line. Our transitional semantics over TCCS then is given by the least relations $\longrightarrow \subseteq$ TCCS $\times$ Act $\times$ TCCS and $\rightsquigarrow \subseteq$ TCCS $\times \mathcal{T} \times$ TCCS (written $P \stackrel{a}{\longrightarrow} Q$ and $P \stackrel{t}{\rightsquigarrow} Q$ respectively) satisfying the rules laid out in Figure 1. Notice that these rules respect the informal description of the constructs given in the previous section.

We can also give operational rules directly for our two derived operators, temporal nil $\underline{0}$ and delayed action prefix $\underline{a}.P$. These are presented in Figure 2. These rules are redundant along with those of Figure 1; however, they will be needed for the subcalculus with which we work later.

We can now define an equivalence relation $\sim$ on closed terms of TCCS based on Park's notion of a *bisimulation* ([Par81]) as follows.

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{P \oplus Q \xrightarrow{a} P'} \qquad \frac{P \xrightarrow{a} P',\ Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{a} P'}{\delta.P \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P \oplus Q \xrightarrow{a} Q'} \qquad \frac{P \xrightarrow{a} P'}{P\backslash L \xrightarrow{a} P'\backslash L}(a, \overline{a} \notin L)$$

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q} \qquad \frac{P \xrightarrow{a} P'}{P[S] \xrightarrow{S(a)} P'[S]}$$

$$\frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'} \qquad \frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'} \qquad \frac{P_i\left\{\mu\tilde{x}.\tilde{P}/\tilde{x}\right\} \xrightarrow{a} P'}{\mu_i\tilde{x}.\tilde{P} \xrightarrow{a} P'}$$

$$\frac{}{\delta.P \overset{t}{\rightsquigarrow} \delta{:}P} \qquad \frac{P \overset{t}{\rightsquigarrow} P',\ Q \overset{t}{\rightsquigarrow} Q'}{P + Q \overset{t}{\rightsquigarrow} P' + Q'} \qquad \frac{P \overset{t}{\rightsquigarrow} P',\ Q \overset{t}{\rightsquigarrow} Q'}{P \mid Q \overset{t}{\rightsquigarrow} P' \mid Q'}$$

$$\frac{}{(s+t).P \overset{s}{\rightsquigarrow} (t).P} \qquad \frac{P \overset{t}{\rightsquigarrow} P'}{P \oplus Q \overset{t}{\rightsquigarrow} P'}(stop(Q) < t) \qquad \frac{P \overset{t}{\rightsquigarrow} P'}{P\backslash L \overset{t}{\rightsquigarrow} P'\backslash L}$$

$$\frac{}{(t).P \overset{t}{\rightsquigarrow} P} \qquad \frac{Q \overset{t}{\rightsquigarrow} Q'}{P \oplus Q \overset{t}{\rightsquigarrow} Q'}(stop(P) < t) \qquad \frac{P \overset{t}{\rightsquigarrow} P'}{P[S] \overset{t}{\rightsquigarrow} P'[S]}$$

$$\frac{P \overset{s}{\rightsquigarrow} P'}{(t).P \overset{s+t}{\rightsquigarrow} P'} \qquad \frac{P \overset{t}{\rightsquigarrow} P',\ Q \overset{t}{\rightsquigarrow} Q'}{P \oplus Q \overset{t}{\rightsquigarrow} P' \oplus Q'} \qquad \frac{P_i\left\{\mu\tilde{x}.\tilde{P}/\tilde{x}\right\} \overset{t}{\rightsquigarrow} P'}{\mu_i\tilde{x}.\tilde{P} \overset{t}{\rightsquigarrow} P'}$$

Figure 1: Operational Rules for TCCS

$$\frac{}{\underline{a}.P \xrightarrow{a} P} \qquad \frac{}{\underline{\mathbf{0}} \overset{t}{\rightsquigarrow} \underline{\mathbf{0}}} \qquad \frac{}{\underline{a}.P \overset{t}{\rightsquigarrow} \underline{a}.P}$$

Figure 2: Operational Rules For Derived Operators

Definition 2.2 *A binary relation $\mathcal{R}$ over terms in* TCCS *is a* $\mathcal{T}$-bisimulation *if and only if for all* $(P, Q) \in \mathcal{R}$ *and for all* $a \in \text{Act}$ *and for all* $t \in \mathcal{T}$,

(i) if $P \xrightarrow{a} P'$ *then* $Q \xrightarrow{a} Q'$ *for some* Q' *with* $(P', Q') \in \mathcal{R}$;

(ii) if $Q \xrightarrow{a} Q'$ *then* $P \xrightarrow{a} P'$ *for some* P' *with* $(P', Q') \in \mathcal{R}$;

(iii) if $P \overset{t}{\rightsquigarrow} P'$ *then* $Q \overset{t}{\rightsquigarrow} Q'$ *for some* Q' *with* $(P', Q') \in \mathcal{R}$;

(iv) if $Q \overset{t}{\rightsquigarrow} Q'$ *then* $P \overset{t}{\rightsquigarrow} P'$ *for some* P' *with* $(P', Q') \in \mathcal{R}$.

$\sim \overset{\text{def}}{=} \bigcup\{\mathcal{R} : \mathcal{R}$ *is a* $\mathcal{T}$*-bisimulation*$\}$ *is then the largest* $\mathcal{T}$*-bisimulation.*

In [Mol90], it is shown that this relation defines a congruence over TCCS terms, and we have also presented there an equational theory which is sound and complete with respect to reasoning about finite-state processes.

2.3 The Problem With Speed

Within the calculus of TCCS, we can describe different *timeout* processes which will allow a certain computation to take place within a given amount of time, but will preempt the computation and proceed with an alternate computation if that amount of time is allowed to pass without the desired computation being performed. As an example, the TCCS term $P + (2).b.\mathbf{0}$ will allow the process P to proceed within 2 units of time, but will subsequently perform the action b and evolve into a deadlocked nil process if the process P does not proceed within the required time. Hence, $P + (2).b.\mathbf{0}$ can be regarded as a timeout context. If we were to replace P with each of the terms $(1).a.\mathbf{0}$ and $(3).a.\mathbf{0}$ which represent the processes which will perform an a action after 1 and 3 units of time respectively, then in the first case, we would result in the process (behaviour) $(1).a.\mathbf{0}$, whereas in the second case we would result in $(2).b.\mathbf{0}$ due to the different timeout scenarios. Clearly we would want to consider the process term $(1).a.\mathbf{0}$ to be *faster than* the process term $(3).a.\mathbf{0}$. However, if we further desired that our *faster than* preorder be a precongruence (that is, that it be substitutive), then we would be led to deduce that the process term $(1).a.\mathbf{0}$ is *faster than* the process term $(2).b.\mathbf{0}$. This is undesirable though, as these two terms are not even functionally behaviourally equivalent.

The problem arises due to the preemptive nature of passing time in the semantics of the operators — it is possible to lose the capability of following a particular computation path through idling. In [Tof90], this problem is challenged by allowing all processes to idle indefinitely without changing state. However, this approach involves an undesirable weakening of the semantic development of the calculus TCCS.

Instead we opt here for a more subtle approach to weakening the expressive power of the calculus. What we actually do is restrict our language to a subset of TCCS where the above undesirable *timeout* notions are no longer definable. We need to insist that, though we can have timeout events made available in the environment after some amount of time, we can never lose the ability to do some other event due to the passing of time. This is not an unreasonable restriction, particularly in the design of hardware circuits, as this is precisely how an implementation behaves — if a port is prepared to communicate, then this communication capability can only be removed through an actual change in state, and not through idling.

3 The Language ℓTCCS

The syntax of the language ℓTCCS (*loose* TCCS) is the subset of TCCS given by the following BNF expression.

$$P ::= \underline{0} \mid X \mid \underline{a}.P \mid (t).P \mid P+P \mid P \mid P \mid P\backslash L \mid P[S] \mid \mu_i \tilde{x}.\tilde{P}$$

Thus we do not have the usual deadlocked nil process $\mathbf{0}$, nor the usual action prefix operator $a.P$ which insists on the action a to be performed without allowing the possibility of any time passing first. Instead we include only the (formerly derived) operators $\underline{0}$ and $\underline{a}.P$. Also, due to the idling nature of actions in ℓTCCS, we no longer need to include the delay operator δ. Indeed we would not like to include this operator in our subcalculus, as a *faster than* relation could not be substitutive with respect to it; where we would clearly have $\underline{a}$ faster than $(1).\underline{a}$, we would not be able to deduce that $\delta.\underline{a}$ was faster than $\delta.(1).\underline{a}$. Finally, in the context of ℓTCCS, the $+$ and $\oplus$ operators behave exactly the same, so we need to include only one of these in the subcalculus.

It is clear now that no temporal deadlocking can occur, so the problem of preempting actions through idling does not arise. This is summarized by the following simple proposition

Proposition 3.1 *For all (closed) ℓTCCS terms P and all $t \in \mathcal{T}$, there is a (unique) P' such that $P \overset{t}{\rightsquigarrow} P'$.*

The semantics is defined as before, using the subset of transition rules given in Figure 1 corresponding to the collection of operators used in the subcalculus, along with the transition rules given in Figure 2 for the temporal nil $\underline{0}$ and delayed action prefix $\underline{a}$. Note that here now the rules of Figure 2 are no longer derivable.

Using the transitional semantics defined here, we can define our *faster than* preorder $\lesssim$ using the following bisimulation-like definition. In this definition, and throughout the sequel, we shall allow ourselves to write the term $(0).P$, and to allow the transition $P \overset{0}{\rightsquigarrow} P$; for $(0)P$ we shall read P, and for $P \overset{0}{\rightsquigarrow} Q$ we shall read *"Q is syntactically identical to P"*.

Definition 3.2 *A binary relation $\mathcal{R}$ over terms of* ℓTCCS *is a* $\lesssim$-bisimulation *if and only if for all* $(P,Q) \in \mathcal{R}$ *and for all* $a \in \mathrm{Act}$ *and for all* $t \in \mathcal{T}$,

(i) if $P \xrightarrow{a} P'$ *then* $Q \overset{s}{\rightsquigarrow} Q' \xrightarrow{a} Q''$ *and* $P' \overset{s}{\rightsquigarrow} P''$ *for some* s, Q', Q'', P'' *with* $\left(P'', Q''\right) \in \mathcal{R}$;

(ii) if $Q \xrightarrow{a} Q'$ *then* $P \xrightarrow{a} P'$ *for some* P' *with* $(P', Q') \in \mathcal{R}$;

(iii) if $P \overset{t}{\rightsquigarrow} P'$ *then* $Q \overset{t}{\rightsquigarrow} Q'$ *for some* Q' *with* $(P', Q') \in \mathcal{R}$;

(iv) if $Q \overset{t}{\rightsquigarrow} Q'$ *then* $P \overset{t}{\rightsquigarrow} P'$ *for some* P' *with* $(P', Q') \in \mathcal{R}$.

$\lesssim \overset{\mathrm{def}}{=} \bigcup\{\mathcal{R} : \mathcal{R}$ *is a* $\lesssim$*-bisimulation*$\}$ *is then the largest* $\lesssim$*-bisimulation.*

Thus the only difference between this definition and the definition of the equivalence given above appears in the first clause: if the first (*faster*) process term can perform a particular action, then the second (*slower*) process term can either perform that action right away and evolve into a new process state which is *slower than* that into which the first process evolved, or else it can idle for some amount of time s and reach a state in which it can perform the action and thus evolve into a state which, while not necessarily itself *slower than* that into which the first process evolved, but *slower than* that state once the idling time is accounted for. As an example, we would clearly want that

$$\underline{a} \mid (1)\underline{b} \lesssim (1)\underline{a} \mid (1)\underline{b}.$$

Now in the *faster* term, the action transition

$$\underline{a} \mid (1)\underline{b} \xrightarrow{a} \underline{\mathbf{0}} \mid (1)\underline{b}$$

is matched in the *slower* term by the sequence of transitions

$$(1)\underline{a} \mid (1)\underline{b} \overset{1}{\rightsquigarrow} \underline{a} \mid \underline{b} \xrightarrow{a} \underline{\mathbf{0}} \mid \underline{b}$$

and while $\underline{\mathbf{0}} \mid (1)\underline{b} \not\lesssim \underline{\mathbf{0}} \mid \underline{b}$, we only require in the definition that $\underline{\mathbf{0}} \mid \underline{b} \lesssim \underline{\mathbf{0}} \mid \underline{b}$, as $\underline{\mathbf{0}} \mid (1)\underline{b} \overset{1}{\rightsquigarrow} \underline{\mathbf{0}} \mid \underline{b}$.

The relation $\lesssim$ is a preorder: it is reflexive, as clearly

$$Id \stackrel{\text{def}}{=} \{(P,P) : P \in \ell\text{TCCS}\}$$

is a $\lesssim$-bisimulation, and it is transitive, as given $\lesssim$-bisimulations $\mathcal{R}_1$ and $\mathcal{R}_2$ we can easily confirm that $\mathcal{R}_3 \stackrel{\text{def}}{=} \mathcal{R}_1\mathcal{R}_2$ is a $\lesssim$-bisimulation.

We can furthermore show that this relation is a precongruence over ℓTCCS terms which do not have any parallel operators within the scope of a recursion. We only demonstrate here the difficult part of the proof of substitutivity, that with respect to the parallel operator.

Proposition 3.3 *Let $\#P$ denote the number of parallel operators in the term P.*

(i) $\mathcal{R}_k = \{(P_1 \mid Q_1, P_2 \mid Q_2) : P_1 \lesssim P_2,\ Q_1 \lesssim Q_2,\ \#(P_i \mid Q_i) \leq k\}$ *is a $\lesssim$-bisimulation.*

(ii) If $\#P \leq k$, $P \stackrel{a}{\longrightarrow} P'$ and $P \stackrel{t}{\rightsquigarrow} P_0$, then $P' \stackrel{t}{\rightsquigarrow} P''$ and $P_0 \stackrel{a}{\longrightarrow} P_0'$ with $P'' \lesssim P_0'$.

Proof: By simultaneous induction on k.

(i) The only difficult clause to check in the definition of $\lesssim$-bisimulation is the first, in the case where the action in question is a synchronization event τ; that is, when $P_1 \mid Q_1 \stackrel{\tau}{\longrightarrow} P_1' \mid P_2'$ by virtue of the fact that $P_1 \stackrel{a}{\longrightarrow} P_1'$ and $Q_1 \stackrel{\overline{a}}{\longrightarrow} Q_1'$. Hence $P_2 \stackrel{s}{\rightsquigarrow} P_2' \stackrel{a}{\longrightarrow} P_2''$, $Q_2 \stackrel{t}{\rightsquigarrow} Q_2' \stackrel{\overline{a}}{\longrightarrow} Q_2''$, $P_1' \stackrel{s}{\rightsquigarrow} P_1''$ and $Q_1' \stackrel{t}{\rightsquigarrow} Q_1''$ with $P_1'' \lesssim P_2''$ and $Q_1'' \lesssim Q_2''$. If $s = t$ then the result follows easily, so we shall assume that $s < t$. By inductive hypothesis *(ii)*, we have that $P_2' \stackrel{t-s}{\rightsquigarrow} \hat{P}_2 \stackrel{a}{\longrightarrow} \hat{P}_2'$ and $P_2'' \stackrel{t-s}{\rightsquigarrow} P_2'''$ with $P_2''' \lesssim \hat{P}_2'$. Furthermore, $P_1'' \stackrel{t-s}{\rightsquigarrow} P_1'''$ with $P_1''' \lesssim P_2'''$, so $P_1''' \lesssim \hat{P}_2'$. Hence $P_2 \mid Q_2 \stackrel{t}{\rightsquigarrow} \hat{P}_2 \mid Q_2' \stackrel{\tau}{\longrightarrow} \hat{P}_2' \mid Q_2''$ and $P_1' \mid Q_1' \stackrel{t}{\rightsquigarrow} P_1''' \mid Q_1''$ with $(P_1''' \mid Q_1'', \hat{P}_2' \mid Q_2'') \in \mathcal{R}_k$.

(ii) Proceeding by structural induction on P, we prove again only the difficult case of parallel composition. Hence assume that $P \mid Q \stackrel{a}{\longrightarrow} P' \mid Q \stackrel{t}{\rightsquigarrow} P'' \mid Q'$ and $P \mid Q \stackrel{t}{\rightsquigarrow} P_0 \mid Q'$ By inductive hypothesis *(ii)*, $P_0 \stackrel{a}{\longrightarrow} P_0'$ with $P'' \lesssim P_0'$. Thus $P_0 \mid Q' \stackrel{a}{\longrightarrow} P_0' \mid Q'$, and then by inductice hypothesis *(i)*, $P'' \mid Q' \lesssim P_0' \mid Q'$. □

The restriction on the parallel operators is there to facilitate our proof; the number of parallel operators in a term could grow through semantic transitions if there were such operators within the scope of a recursion. We conjecture that this restriction can be removed, but have failed to provide a proof of this more general result.

It is clear from the similarities in the definitions of $\sim$ and $\lesssim$ that for P and Q being two terms of ℓTCCS, if $P \sim Q$ then $P \lesssim Q$ (and $Q \lesssim P$). However the reverse implication does not hold; that is, $P \lesssim Q$ and $Q \lesssim P$ does not necessarily imply that $P \sim Q$. A suitable simple counter-example is provided by the following two process terms:

$$A \stackrel{\text{def}}{=} \underline{a}\underline{b} + \underline{a}(1)\underline{b} + \underline{a}(2)\underline{b} \qquad\qquad B \stackrel{\text{def}}{=} \underline{a}\underline{b} + \underline{a}(2)\underline{b}$$

These two processes are equally fast by the above definition, yet are not equivalent, as $A \stackrel{a}{\longrightarrow} (1)\underline{b}$, but for no $B' \sim (1)\underline{b}$ does $B \stackrel{a}{\longrightarrow} B'$. Hence another equivalence of interest is $\lesssim \cap \gtrsim$, which we represent by $\cong$.

We have another general anomaly with this — or indeed any — *faster than* preorder for nondeterministic processes: we cannot guarantee that if $P \lesssim Q$, then P will *necessarily* execute faster than Q, but only that it has the capability of so doing. We would for example insist by reflexivity that for the above process A, $A \lesssim A$; but in executing the two instances of A, the first (supposedly *faster*) version may start with an a transition to the state $(2)\underline{b}$, whereas the second version may start with an a transition to the state $\underline{b}$. However, this problem only arises in the presence of nondeterminism, and also vanishes if we assume some form of built-in priority allowing faster computation paths to be followed whenever possible.

3.1 Example

My sister Velma in Canada subscribes to Maclean's, "Canada's weekly newsmagazine". When she is finished reading these, she duly sends them to me to keep me informed of the latest developments on the Canadian scene. In fact, she *always* sends me 6 issues at a time, so that a new package gets sent approximately every 42 days (it takes that long to amass 6 copies, but my sister sometimes procrastinates and doesn't put them into an envelope, or doesn't go to the post office with them, until some indeterminate time later).

When she started this practice six years ago, she always sent the magazines by air mail. Canada Post would receive the package from her, and deliver it about four days later; it always takes at least four days to deliver an air mail package from Canada to Scotland, but it could take substantially much longer still. Thus the behaviour exhibited by Canada Post in delivering the i^{th} package is represented by the following equation:

$$\text{AM}_i \stackrel{\text{def}}{=} \underline{\text{in}_i}.(4).\overline{\underline{\text{del}_i}}.\mathbf{0}.$$

That is, the post office would receive the i^{th} package at some point in time ($\underline{\text{in}_i}$), then not communicate with the external environment (myself and my sister) for four days, after which (and this means at some indeterminate time afterwards) it would deliver the package to me ($\overline{\underline{\text{del}_i}}$). Thus ended the process of delivering the i^{th} package.

The behaviour of the post office with respect to my magazine packages then was dictated by the following equation:

$$\mathrm{RS}_i \stackrel{\text{def}}{=} \mathrm{AM}_i \mid (42).\mathrm{RS}_{i+1}.$$

That is, my *Rich Sister* RS would be ready to send the i^{th} package by air mail, while at the same time, another 42-day period was being counted down before the next package was ready to send.

There are interesting properties associated with this process, which are inherent in the loose semantic notion of timing. Firstly, of course I could not expect to receive the i^{th} package of magazines before $42i$ days had passed; I could not read the news before it was written! But apart from that, I could not know with any certainty when I would receive my package; I was dependent on the reliability of my sister and of Canada Post. In fact I could receive the packages in the wrong order (in fact in an arbitrary order), which on occasion did happen.

Alas, my sister quickly felt the squeeze as the expense of sending regular air mail packages became too much, and she started sending the packages occasionally by surface mail. Thus her behaviour started being dictated by the following equation:

$$\mathrm{MS}_i \stackrel{\text{def}}{=} (\mathrm{AM}_i + \mathrm{SM}_i) \mid (42).\mathrm{MS}_{i+1},$$

where SM_i is the behaviour of Canada Post when handling a package being sent by surface mail:

$$\mathrm{SM}_i \stackrel{\text{def}}{=} \underline{\mathrm{in}}_i.(4).\overline{\underline{\mathrm{del}}}_i.\mathbf{0} + \underline{\mathrm{in}}_i.(60).\overline{\underline{\mathrm{del}}}_i.\mathbf{0}.$$

Packages sent by surface mail take at least 60 days to be delivered, but on regular occasions a package which is posted by surface mail is thrown into an airmail bag and is delivered at airmail speed.

Hence my *Middle-class Sister* MS saved money by occasionally sending me my magazines by surface mail. The change in the system was clearly noticeable, as I found myself reading even older news. However, it was still news to me, as I have no other reliable source for Canadian current events. As I never complained to her, and since she still felt the budget crunch whenever she sent a pakage of magazines to me by air mail, my sister eventually quit sending packages by air mail altogether. Her behaviour then became dictated by the following equation:

$$\mathrm{PS}_i \stackrel{\text{def}}{=} \mathrm{SM}_i \mid \mathrm{PS}_{i+1}.$$

Henceforth, my *Poor Sister* PS sent me all my magazines by surface mail. However, I noticed the change much less so than with the original change, as I came to expect my magazines to arrive late, and was only surprised when they arrived early. This situation still remains, as my magazines regularly arrive by air mail post, though they were intended by my sister to be sent by surface mail.

The differences in the delivery times, and the fact that I only really noticed the effect of the first change, can be seen from the faster-than relationships exibited in the above processes. Firstly we have that

$$\mathrm{AM}_i \lesssim \mathrm{AM}_i + \mathrm{SM}_i \lesssim \mathrm{SM}_i,$$

and in fact more than that we have that

$$\mathrm{RS}_i \lesssim \mathrm{MS}_i \lesssim \mathrm{PS}_i.$$

Thus I certainly couldn't expect to notice a speedup in the delivery of my magazines. However, note also that

$$\mathrm{SM}_i \lesssim \mathrm{AM}_i + \mathrm{SM}_i \not\lesssim \mathrm{AM}_i,$$

and in fact

$$\mathrm{PS}_i \lesssim \mathrm{MS}_i \not\lesssim \mathrm{RS}_i,$$

That is, the magazines sent by my *rich* sister were delivered *as fast as* those sent by my *middle-class* sister, but not vice-versa; whereas those sent by my *middle-class* sister were delivered *as fast as* those sent by my *poor* sister, **and vice versa**. Hence I never noticed this second change the way that I did the first. If however, Canada Post reliably sent the surface mail packages by surface mail, then I would never receive any early packages, and the resulting system would *not* be *as fast as* $\mathrm{AM}_i + \mathrm{SM}_i$, and the difference would be noticeable,

4 Inequational Laws

In this section, we develop a set of inequational laws for ℓTCCS which are sound with respect to our semantic precongruence $\lesssim$. We shall use $\leq$ to represent derivability in the theory (along with $=$ representing $\leq \cap \geq$), and $\equiv$ to represent syntactic identity.

$(+_1)$	$(x+y)+z = x+(y+z)$	$(+_2)$	$x+y = y+x$
$(+_3)$	$x+x = x$	$(+_4)$	$x+\underline{0} = x$
(T_1)	$(s)(t)x = (s+t)x$	(T_2)	$(t)x+(t)y = (t)(x+y)$
(T_3)	$x+(t)x = x$	(T_4)	$(t)\underline{0} = \underline{0}$
(T_5)	$x \leq (t)x$		

Figure 3: Equational Theory for Sequential ℓTCCS

4.1 Finite Sequential Terms

We start our analysis by restricting our attention to the finite sequential subcalculus of ℓTCCS given by the following BNF equation.

$$P \ ::= \ \underline{0} \ \mid \ \underline{a}.P \ \mid \ (t).P \ \mid \ P+P$$

We shall refer to this subcalculus by ℓTCCS_0.

Our inequational theory for ℓTCCS_0 is presented in Figure 3. We can straightforwardly demonstrate that these laws are valid with respect to our semantic precongruence $\lesssim$ over the full calculus ℓTCCS. Hence we get the following result.

Proposition 4.1 (Soundness) $p \leq q \implies p \lesssim q$.

Next we want to show that these laws are complete for reasoning over ℓTCCS_0. We accomplish this using the following normal form.

Definition 4.2 *A term is a NF term (or is in NF) if it is of the form*

$$\sum_{1 \leq i \leq m} (t_i).\underline{a}_i.p_i$$

where each p_i is itself in NF.

Our first task in proving completeness is to show that every term in ℓTCCS can be converted into *NF* using these laws.

Proposition 4.3 *Every term $p \in \ell\text{TCCS}_0$ can be equated to a term in NF using the laws of* Figure 3.

Proof: A straightforward structural induction on p. □

So for our completeness result, we only need to show that for *NF* terms P and Q, if $P \lesssim Q$ then $P \leq Q$.

Proposition 4.4 *If* $P \lesssim Q$ *where* $P \equiv \sum_i (s_i).\underline{a}_i.p_i$ *and* $Q \equiv \sum_j (t_j).\underline{b}_j.q_j$ *are in NF, then* $P \leq Q$.

Proof: Let P and Q be given as above. It suffices to prove the following facts.

(i) $\forall i \exists j : a_i = b_j \ \wedge \ p_i \lesssim q_j$;

(ii) $\forall j \exists i : a_i = b_j \ \wedge \ p_i \lesssim q_j \ \wedge \ s_i \leq t_j$.

From the first fact, using structural induction, we can deduce that

$$(s_i).\underline{a}_i.p_i + (t_j).\underline{b}_j.q_j \ \leq \ (t_j).\underline{b}_j.q_j,$$

from which we can deduce that $P + Q \leq Q$; and from the second fact, again using structural induction, we can deduce that

$$(s_i).\underline{a}_i.p_i \leq (t_j).\underline{b}_j.q_j,$$

from which we can deduce that $P \leq P + Q$. Putting these together, we get $P \leq Q$.

Both these facts follow immediately from the definition of $\lesssim$, noting that

$$\sum_i (s_i \dot{-} t).\underline{a}_i.p_i \lesssim \sum_j (t_j \dot{-} t).\underline{b}_j.q_j$$

where $x \dot{-} y = \max(0, x - y)$. □

4.2 Adding Concurrency

ℓTCCS, like CCS, is an *interleaving* theory of concurrency. That is, we can represent the parallel functional behaviour of processes in terms of causality and nondeterministic choice. The typical example is given by the equation

$$\underline{a} \mid \underline{b} \ = \ \underline{ab} + \underline{ba}$$

This equation is an instance of what is known as the *Expansion Theorem.* A major advantage of this treatment of concurrency is that the algebraic analysis of concurrent terms can be reduced to the analysis of sequential terms.

Unfortunately, when we add timing prefixes, our analysis becomes complicated, as the expressive power of the calculus becomes inadequate to express parallel processes as equivalent sequential terms. In the calculus TCCS, this problem was successfully overcome by introducing the weak choice operator $\oplus$. However, we cannot find such a sensible extension to the calculus ℓTCCS for which the notion of *faster than* remains substitutive, to allow ourselves to express every (finite) term in an equivalent sequential form. For example, the term $\underline{a} \mid (1)\underline{b}$ has no equivalent sequential form. We would want to be able to express this term by its expansion, namely as $\underline{a}(1)\underline{b} + (1)(\underline{a} \mid \underline{b})$ (where we can then recursively expand the subterm $\underline{a} \mid \underline{b}$). However this term is clearly not equivalent to the original, as the expanded term can idle one unit of time, then perform an a action and evolve into a state where it must idle for one more unit of time before being capable of performing the b action, whereas the parallel term after idling one unit of time and then performing the a action, will always be capable of immediately performing the b action.

Though the usual expansion of a parallel term is not necessarily equivalent to the term itself, we do have the result that it is related in one direction to the parallel term in the *faster than* relation; the parallel term is guaranteed to be *faster than* the sequentialised expanded term. These expansion principles are presented in Figure 4.

Hence, we thus cannot use the usual technique in comparing two terms of expressing the terms as equivalent sequential terms and then comparing these sequential terms using our complete set of laws for such terms. However, the expansion of a (parallel) term is so very close to being equivalent to the term itself, that we conjecture that it is not possible to find a term which falls in the *faster than* relation strictly between a term and its expanded version.

In [Mol90], we introduced valid Expansion Theorem laws for the full calculus TCCS which are equally valid here if we do not use substitutivity as a proof rule. These laws could be used as well in a proof as long as due care were taken.

5 Related Work

There have been many descriptions of process algebras as the foundation of concurrency theory, but only recently have any of these models been extended to consider the problem of describing the real-time

Let $X = \sum_{1\le i\le m} \underline{a}_i x_i$ and $Y = \sum_{1\le j\le n} \underline{b}_j y_j$

$$(E_1)\quad X \mid Y = \sum_{1\le i\le m} \underline{a}_i(x_i \mid Y) + \sum_{1\le j\le n} \underline{b}_j(X \mid y_j) + \sum_{\underline{a}_i=\overline{b}_j} \underline{\tau}(x_i \mid y_j)$$

$$(E_2)\quad X \mid (Y+(1)y) \le \sum_{1\le i\le m} \underline{a}_i(x_i \mid (Y+(1)y)) + \sum_{1\le j\le n} \underline{b}_j(X \mid y_j) + \sum_{\underline{a}_i=\overline{b}_j} \underline{\tau}(x_i \mid y_j) + (1)(X \mid (Y+y))$$

$$(E_3)\quad (X+(1)x) \mid Y \le \sum_{1\le i\le m} \underline{a}_i(x_i \mid Y) + \sum_{1\le j\le n} \underline{b}_j((X+(1)x) \mid y_j)) + \sum_{\underline{a}_i=\overline{b}_j} \underline{\tau}(x_i \mid y_j) + (1)((X+x) \mid Y)$$

$$(E_4)\quad (X+(1)x) \mid (Y+(1)y) \le \sum_{1\le i\le m} \underline{a}_i(x_i \mid (Y+(1)y)) + \sum_{1\le j\le n} \underline{b}_j((X+(1)x) \mid y_j)) + \sum_{\underline{a}_i=\overline{b}_j} \underline{\tau}(x_i \mid y_j) + (1)(x \mid y) + (1)((X+x) \mid (Y+y))$$

Figure 4: Expansion Laws

aspects of concurrent systems. Especially recently, there has been a spate of developments involving each of the main process algebra approaches aimed at incorporating real-time aspects into the theory. As a sample we refer to [Ree86, Bae89, Gro90, Hen90, Mol90, Nic90, Wan90].

None of these previous attempts have considered the problem of relating process terms with respect to speed. Most of the approaches cited above allow for actions to be made to occur at specified times (as with the calculus TCCS described in this report and in [Mol90]), and thus would not allow for a substitutive notion of a relative speed relation. Others, including [Hen90, Wan90], are similar to ℓTCCS in that they only specify the delay before a particular action may occur, allowing the action to occur any time after that delay; these approaches would then have difficulty with the axiomatization of concurrency as described above, with the theory of process equality as well as with a *faster than* relation. Also, these two particular approaches include a *maximal progression* principle which enforces communications to occur at the time which they may occur without delay, thus returning to the problems of defining such a desired substitutive relation.

There has been one other approach to describing relative efficiency in process algebras, described in

[Aru90]. This relation equates two process terms if they are bisimilar with the proviso that the faster term need never perform more internal τ actions than the slower term. This approach does not deal with real-time issues in process algebra, but does go a long way towards analysing the types of questions of interest in the notion. A similar relation, coined a *contraction* by Milner, is presently implemented in the Edinburgh Concurrency Workbench, and is described in [CWB90].

Bibliography

[Aru90] Arun-Kumar, S., M. Hennessy, *An Efficiency Preorder for Processes*, University of Sussex Research Report No. 5/90, 1990.

[Bae89] Baeten, J.C.M., J.A. Bergstra, *Real Time Process Algebra*, Preliminary Draft, 10/20/89, 1989.

[CWB90] *The Edinburgh Concurrency Workbench – Operating Instructions*, University of Edinburgh Technical Report, 1990.

[Gro90] Groote, J.F., *Specification and Verification of Real Time Systems in ACP*, Research Report No CS-R9015, Centre for Mathematics and Computer Science, Amsterdam, 1990.

[Hen90] Hennessy, M., T. Regan, *A Temporal Process Algebra* Technical Report No. 2/90, University of Sussex Computer Science Department, April, 1990.

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

[Mil83] Milner, R., *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science, Vol 25, 1983.

[Mil89] Milner, R., **Communication and Concurrency**, Prentice–Hall International, 1989.

[Mol90] Moller, F., C. Tofts, *A Temporal Calculus of Communicating Systems*, Proceedings of CONCUR'90 (Theories of Concurrency: Unification and Extension), Amsterdam, August 1990.

[Nic90] Nicollin, X., J.L. Richier, J. Sifakis, J.Voiron, *ATP: An Algebra for Timed Processes*, Proceedings of IFIP Working Conference on Programming Concepts and Methods, North Holland, 1990.

[Par81] Park, D.M.R., *Concurrency and Automata on Infinite Sequences*, Lecture Notes in Computer Science 104, Springer–Verlag, 1981.

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

[Ree86] Reed, G.M., A. Roscoe, *A Timed Model for Communicating Sequential Processes*, Proceedings of ICALP'86, Lecture Notes in Computer Science No 226, Springer Verlag, 1986.

[Tof90] Tofts, C., *Proof Systems and Pragmatics for Parallel Programming*, PhD Thesis, University of Edinburgh, 1990.

[Wan90] Wang Yi, *Real-time Behaviour of Asynchronous Agents*, Proceedings of CONCUR'90 (Theories of Concurrency: Unification and Extension), Amsterdam, August 1990.