CCS + Time = an Interleaving Model for Real Time Systems

Wang Yi *

Abstract

This paper shows how to put time into Milner's CCS to model real time systems. In particular, we will develop an expansion theorem for real time concurrency, which is an extension of the expansion theorem of CCS. The essential step made in this work is that a more general form of action prefix, $\mu@t.P$ is introduced, where t is a time variable. Intuitively, $\mu@t.P$ is an agent which may perform μ and become $P[d/t]$ in doing so, where t is replaced by d, the time delay before μ is actually performed. The original form $\mu.P$ of action prefix of CCS is just a simple case of $\mu@t.P$ when t does not occur free in P — P does not depend on the time at which μ is performed.

1 Introduction

The aim of this work is to develop an interleaving model for "real time" systems by extending Milner's CCS, [M89] with real time. The model is interleaving in the sense that concurrency is interpreted in terms of non-determinism. This fact shall be expressed by an expansion theorem as in CCS.

The first question arises: what is a real time system? Instead of finding an accurate definition to encompass all the characteristics of real time systems, we shall use the term "real time" to describe the class of systems that have to respond to externally generated stimuli or inputs within specified time limits. Also, *the following behaviour of such a system depends on the time at which the external stimuli arise.* This type of systems includes many safety-critical systems such as robotics and flight control systems. We will use agents to model actual systems and the agents are built out of CCS-like operators. To deal with real time, CCS action prefix shall be extended to the form $\mu@t.P$ and moreover, a delay-construct, $\epsilon(d).P$ in prefix form is introduced. We wish that this should be the least extension to CCS to achieve an interleaving timed model.

Unlike most of the other timed models based on CCS such as [HR90,S90,MT90] which only deal with discrete time, our model is independent of a time domain as long as it is a numerical domain such as the natural numbers. For convenience, in this paper we have only considered the case when the time domain is the set of positive reals including 0. However, it can also deal with other time domains such as the rational numbers and natural numbers. In particular, the natural numbers shall yield a discrete timed CCS which includes Hennessy and Regan's TPA [HR90] as a subcalculus and the singleton $\{0\}$ shall give us the untimed calculus, CCS [M89]. In fact, the calculus is essentially independent of a time domain as long as it is a totally ordered monoid, [J91].

The main results of this paper include the following. Strong and weak equivalences for timed agents are defined based on the well established notion of bisimulation on labelled transition systems. Strong equivalence is a congruence and weak equivalence is not, but preserved by all operators except summation and recursion. A number of equational laws are developed. In particular, an expansion theorem is presented, which is an extension of CCS expansion theorem.

The paper is organized as follows. Section 2 motivates some important design decisions including the maximal progress and time determinacy assumptions. Section 3 describes the syntax and

*Address: Department of Computer Sciences, University of Göteborg and Chalmers University of Technology, S–412 96 Göteborg, Sweden. E–mail: yi@cs.chalmers.se

operational semantics for the calculus, timed CCS. Section 4 establishes bisimulation equivalences. Section 5 develops algebraic laws in terms of bisimulation equivalences. In particular, an expansion theorem is presented. Section 6 presents a brief comparison with related work.

2 Design Decisions

2.1 From $\lambda x.f$ to $\alpha x.P$ and $\alpha x@t.P$

Recall the meaning of a program described as λ-term, $\lambda x.f$. The program expects an input and when the input data v is given, it computes the value of $f[v/x]$ as output. This has motivated the construct $\alpha x.P$ of the value-passing CCS. Similarly, $\alpha x.P$ describes an agent which is waiting for a message on channel α; when a message v is available on α, the agent will behave like $P[v/x]$.

Note that the following behaviour of $\alpha x.P$ depends on the content v of the received message, but not the time when v is available. As emphasized earlier, the following behaviour of a real time system depends on the time *when* an external stimulus such as a message arises. Therefore, the construct $\alpha x.P$ is not adequate for modelling such systems. To meet this requirement, we introduce an extra variable t recording the time delay before a message on α is available. We use $\alpha x@t.P$ to stand for an agent which is waiting for a message on channel α. When a message arrives, the agent will behave like $P[v/x, d/t]$ where

- v is the *content* of the message and
- d is the *delay* before the message is available.

Note that the timing information about an incoming message is also considered as a part of input. We extend the idea to the construct $\bar{\alpha}v.P$ of CCS and use $\bar{\alpha}v@t.P$ to stand for an agent which is to deliver a message v on channel α. When the message is delivered, the agent will behave like $P[d/t]$ where

- d is the *delay* before message v is delivered.

When the content of a received message does not affect the future behaviour of an agent, it shall be ignored. This is called pure synchronization in CCS. We use

$$\alpha@t.P$$

to denote an agent which is waiting for the environment to synchronize on α and then become $P[d/t]$ in doing so where d represents the delay before the synchronization on α takes place.

In many cases, the following behaviour of a system does not depend on the content of a received message, nor the time at which the message arrives. The pure CCS captures this aspect. As usual in CCS, we use $\alpha.P$ to describe an agent which is waiting for the environment to synchronize on α and then becomes P in doing so. Here, $\alpha.P$ can be understood as a notation for $\alpha@t.P$ when P does not depend on variable t — the following agent P does not depend on the time at which the synchronization on α takes place. Hence, $\alpha@t.P$ is a generalization of the original action prefix $\alpha.P$ of CCS.

We end this subsection with an abstract description of a simple stop watch to illustrate the potential application of the new construct $\alpha@t.P$ in modelling real time behaviour. We will use the more illustrative notation $\alpha!v$ instead of $\bar{\alpha}v$.

$$\begin{aligned} S &\stackrel{def}{=} start.G \\ G &\stackrel{def}{=} stop@t.D(t) \\ D(t) &\stackrel{def}{=} display!t.S \end{aligned}$$

The timer is always ready to accept a *start*-signal and then wait for the *stop*-signal. When the *stop*-signal arises, the time delay between *start* and *stop* is displayed.

2.2 From *delay*(d); P to $1.P$ and $\epsilon(d).P$

Many programming languages such as Ada, Esterel and Occam, provide delay operators to suspend activities. For example, Ada has a construct, *delay*(d); P which means "wait for d units of time and then execute P".

In SCCS, $1.P$ is an agent that idles one unit of time and then behaves like P. Similarly, we use symbol ϵ to represent idling and write $\epsilon(d).P$ to denote an agent which idles for d units of time and then behaves like P. Note that d can be any positive number. This construct is rather classical. Its uses in modelling real time systems need no further emphasis.

2.3 Actions, Instantaneousness and Controllability

Instantaneousness of Actions: We take the view that atomic actions are instantaneous. For example, if the environment is ready to offer $\bar{\alpha}$, then $\alpha.P$ will perform α immediately and become P in doing so. But whenever it is necessary, we can put a delay before an action using the delay-construct. The point is that we want to separate the time delay for enabling an atomic action and the moment at which the action is *committed.* For example, it will take 3 units of time for $\epsilon(3).\alpha.P$ to be ready to offer α. This approach has been taken by other work such as *ATP*, [S90] and Esterel. Furthermore, we distinguish two types of atomic actions: *controllable* and *non-controllable.*

Controllable Actions: They are also called visible actions in CCS. A controllable action α needs the external stimulus $\bar{\alpha}$ to occur. So an agent can never complete a controllable action by itself. However, by offering $\bar{\alpha}$, an agent can influence another agent which is capable of doing α. We write $P \xrightarrow{\alpha} Q$ to mean that *if the environment is ready to offer* $\bar{\alpha}$, P *may perform* α *immediately and become* Q *in doing so.* This transition takes no time, because α is instantaneous.

Non-Controllable Actions: They are also called invisible actions in CCS. In contrast with a controllable action, a non controllable action is autonomous and does not require external stimulus to occur. A typical non-controllable action is an internal communication between two agents which perform α and $\bar{\alpha}$ simultaneously. As in CCS, we use τ to represent non controllable actions and write $P \xrightarrow{\tau} Q$ to mean that P *may perform* τ *autonomously and become* Q *in doing so.* Note that the environment is not required to participate in this transition. This transition takes no time either, because τ is instantaneous.

2.4 Idling, Maximal Progress and Time Determinacy

Idling: The central idea of real time systems is to study whether a system is able to perform a certain action within a given time limit. During the time when no action is performed, the behaviour of a system is considered to be *idling.*

Recall that in SCCS, $P \xrightarrow{1} Q$ means that if P exists at time r, it will proceed to Q at time $r+1$. The time delay is exactly one unit. We extend this idea to an arbitrary time delay. As suggested earlier, symbol ϵ denotes idling. We write

$$P \xrightarrow{\epsilon(d)} Q$$

to mean that P *will proceed to* Q *in* d *units of time.* Unlike $P \xrightarrow{\alpha} Q$ and $P \xrightarrow{\tau} Q$ which take no time, *this transition takes* d *units of time.* For instance,

1. $\epsilon(3).P \xrightarrow{\epsilon(2.5)} \epsilon(0.5).P$ means that $\epsilon(3).P$ proceeds to $\epsilon(0.5).P$ in 2.5 units of time and

2. $\epsilon(3).P \xrightarrow{\epsilon(3)} P$ means that $\epsilon(3).P$ proceeds to P in 3 units of time.

As in CCS, a controllable (visible) action always waits for the environment to synchronize. For example, we shall have $\alpha.P \xrightarrow{\epsilon(d)} \alpha.P$ for all $d > 0$. It is waiting for $\bar{\alpha}$ which shall be offered by the environment.

Now, two types of waiting have been introduced. First, $\epsilon(d).P$ is forced to wait for d units of time. By contrast, $\alpha.P$ is willing to wait, if the environment is not ready to offer $\bar{\alpha}$. However, we shall not distinguish these two types of waiting, because they are different only in an intensional sense. No environment shall be able to discover the difference externally. Hence, the transition $P \xrightarrow{\epsilon(d)} Q$ is internal in the sense that P will have no information exchange with the environment for d units of time.

Maximal Progress: We have introduced three types of transitions:

1. $P \xrightarrow{\alpha} Q$ takes no time and requires the environment to participate in.
2. $P \xrightarrow{\tau} Q$ takes no time and is autonomous (independently of the environment).
3. $P \xrightarrow{\epsilon(d)} Q$ takes d units of time and is internal (no interaction with the environment).

An agent may have several different types of transitions. For example, $\alpha.P \xrightarrow{\alpha} P$ and also $\alpha.P \xrightarrow{\epsilon(d)} \alpha.P$. The second transition means that $\alpha.P$ is *waiting for the environment.* Now, the question arises: whether $\tau@t.P$ is allowed to wait. A more general question is whether an agent is allowed to wait, when it can perform τ *independently* of the environment. Suppose that the answer is "yes". Then what is $\tau@t.P$ waiting for and how long time it is allowed to wait? Our decision is that *if an agent can proceed by performing τ-actions, then it should never wait unnecessarily.* This is called *maximal progress* assumption. It simply requires that for each agent P, whenever $P \xrightarrow{\tau} P'$ for some P', then $P \xrightarrow{\epsilon(d)} P''$ for no d and P''. We will discover that the maximal progress assumption determines the whole design of our calculus. To ensure this assumption, we have only one rule for τ-actions:

$$\boxed{\tau@t.P \xrightarrow{\tau} P[0/t]}$$

As mentioned earlier, τ-transition takes no time and therefore t is replaced by 0 in the following agent. For controllable actions, there are two rules:

$$\boxed{\begin{array}{ll} (1) & \alpha@t.P \xrightarrow{\alpha} P[0/t] \\ (2) & \alpha@t.P \xrightarrow{\epsilon(d)} \alpha@t.P[t+d/t] \end{array}}$$

By the first rule, if $\bar{\alpha}$ is available, $\alpha@t.P$ shall perform α immediately and become $P[0/t]$ in doing so. This transition takes no time and therefore t is replaced by 0 in the following agent. By the second rule, if $\alpha@t.P$ has waited d units of time for $\bar{\alpha}$, then t should be increased by d. For instance, we have $\alpha@t.\epsilon(t).\beta \xrightarrow{\epsilon(3)} \alpha@t.\epsilon(t+3).\beta \xrightarrow{\alpha} \epsilon(3).\beta$.

Time Determinacy: Usually in CCS, an agent may reach different states non-deterministically after performing an atomic action. We assume that non-deterministic choice can only be made by actions, not idling. This is another significant design decision, called *time determinacy.* Formally, it requires that whenever $P \xrightarrow{\epsilon(d)} P_1$ and $P \xrightarrow{\epsilon(d)} P_2$, then P_1 and P_2 are identical.

For example, let $M \stackrel{def}{=} coffee!.M + tea!.M$, where M describes a vending machine which is always ready to deliver a drink. Obviously, we would like to have

$$coffee!.M + tea!.M \xrightarrow{\epsilon(d)} coffee!.M + tea!.M$$

for all $d > 0$.

The machine is waiting for the user. This motivates the following rule for summation,

$$\frac{P \xrightarrow{\epsilon(d)} P' \qquad Q \xrightarrow{\epsilon(d)} Q'}{P+Q \xrightarrow{\epsilon(d)} P'+Q'}$$

The rule embodies the time determinacy and also maximal progress assumptions. For example, $\alpha.P + \epsilon(3).\tau.Q$ can only wait 3 units of time for $\bar{\alpha}$ and then it will perform τ immediately and become Q in doing so. Here, τ may be understood as an internal timeout event.

3 Timed CCS

3.1 Syntax and Semantics

Given a time domain $\mathcal{T}$ with a least element 0, let $\delta_{\mathcal{T}}$ be the set $\{\epsilon(d) | d \in \mathcal{T} - \{0\}\}$ of time events. Note that $\delta_{\mathcal{T}}$ does not include $\epsilon(0)$. Each $\epsilon(d)$ of $\delta_{\mathcal{T}}$ may be considered as an empty action which lasts for d units of time. Here, the time domain shall be a numerical domain such as the natural numbers. For convenience, in this paper, we shall only consider the case when $\mathcal{T}$ is $\mathcal{R}^{+0}$, the set of the positive reals including 0.

We shall use c, d to range over $\mathcal{R}^{+}$, the positive reals. Assume a set of time variables ranged over by t, u, x, y. We build time expressions such as $c \dot{-} d$ and $t + d$, out of the constants $\mathcal{R}^{+0}$, variables and binary operations $+$ and $\dot{-}$, where $\dot{-}$ is defined by $c \dot{-} d = 0$ if $d \geq c$, otherwise $c - d$. We use e as meta-variable to range over time expressions.

Let Λ denote the set of controllable actions and τ represent non controllable actions. Following Milner, let $\Lambda = \Delta \cup \bar{\Delta}$, where Δ is known as the set of names and $\bar{\Delta} = \{\bar{\alpha} | \alpha \in \Delta\}$ is the set of co-names. The action $\bar{\alpha}$ is the stimulus or complement of α. Also, α shall be viewed as the stimulus of $\bar{\alpha}$, i.e. $\bar{\bar{\alpha}} = \alpha$. Note that τ and $\epsilon(d)$ have no complements. However for convenience, we define $\bar{\tau} = \tau$ and $\overline{\epsilon(d)} = \epsilon(d)$.

Let $\mathcal{A}ct = \Lambda \cup \{\tau\}$ and $\mathcal{L} = \mathcal{A}ct \cup \delta_{\mathcal{R}^{+0}}$. We use α, β to range over Λ and μ, ν to range over $\mathcal{A}ct$ and σ to range over $\mathcal{L}$. For the purpose of restriction and relabeling, let L be a subset of Λ and S be a bijection from $\mathcal{A}ct$ to $\mathcal{A}ct$ with $S(\epsilon(d)) = \epsilon(d)$, $S(\tau) = \tau$ and $S(\bar{\alpha}) = \overline{S(\alpha)}$.

Further, assume a set of agent variables ranged over by X. Then the language has the following BNF-grammar.

$$E ::= \text{NIL} \mid X \mid \epsilon(e).E \mid \mu@t.E \mid E + F \mid E|F \mid E\backslash L \mid E[S] \mid \boldsymbol{rec}X : E$$

Observe that the syntax of TCCS is essentially the same as CCS except that the action prefix of CCS has been extended to the form $\alpha@t.E$ and a delay construct $\epsilon(e).E$ has been introduced. It is important to note that expressions generated by the above grammar may contain two kinds of variables: time variables and agent variables. Time variables are bound with $\mu@$ where $\mu \in \mathcal{A}ct$; agent variables are bound with ***rec***. As usual, a bound variable can be replaced with another variable and therefore we shall not distinguish $\mu@t.E$ from $\mu@u.E[u/t]$. This shall be justified semantically later. When E does not depend on time variable t, we shall write $\mu.E$ instead of $\mu@t.E$.

A closed expression containing no free variables is called an *agent*. We use $\mathcal{P}_R$ to denote the set of agents, ranged over by P, Q, R.

The transition rules for $\mathcal{P}_R$ are given in table 1. Note that the side condition for the timing rule of parallel composition is to guarantee that the composite agents satisfy the maximal progress assumption. Intuitively, $\mathcal{S}ort_d(P) \cap \overline{\mathcal{S}ort_d(Q)} = \emptyset$ means that P and Q cannot communicate with each other within d units of time. Note also that $\mathcal{S}ort$ is defined on syntactical terms independently of the transition relation $\longrightarrow$.

Given a time interval d, $Sort_d(P)$ shall include all controllable actions which P may perform within d units of time; but there may be some action in $Sort_d(P)$ that P cannot perform at all because of the maximal progress assumption. For example, $Sort_3(\tau.\mathrm{NIL} + \epsilon(2).\alpha.P) = \{\alpha\}$; but $\tau.\mathrm{NIL} + \epsilon(2).\alpha.P$ will perform τ immediately and then become NIL.

Definition 1 *Let E be an agent expression containing no free time variable. Given an assignment function ρ assigning each free agent variable a subset of Λ, we define $Sort_0(E)\rho = \emptyset$ and $Sort_d(E)\rho$ to be the least set satisfying:*

$$
\begin{array}{rcl}
Sort_d(X)\rho & = & \rho(X) \\
Sort_d(NIL)\rho & = & \emptyset \\
Sort_d(\alpha@t.E)\rho & = & \{\alpha\} \\
Sort_d(\tau@t.E)\rho & = & \emptyset \\
Sort_d(\epsilon(e).E)\rho & = & Sort_{d \dot{-} e}(E)\rho \\
Sort_d(E+F)\rho & = & Sort_d(E)\rho \cup Sort_d(F)\rho \\
Sort_d(E|F)\rho & = & Sort_d(E)\rho \cup Sort_d(F)\rho \\
Sort_d(E\backslash L)\rho & = & Sort_d(E)\rho - \{L \cup \bar{L}\} \\
Sort_d(E[S])\rho & = & \{S(\alpha) | \alpha \in Sort_d(E)\rho\} \\
Sort_d(\mathbf{rec}X : E)\rho & = & \mu S : H(S)
\end{array}
$$

where $H(S) \equiv Sort_d(E)\rho_{[X \mapsto S]}$ and $\mu S : H(S)$ is the least fixed point of function H.

For an agent P (closed expression), we shall omit the assignment function ρ and write $Sort_d(P)$ directly. For example, $Sort_2(\epsilon(1.5).\alpha.P) = \{\alpha\}$ and $Sort_3(\mathbf{rec}X : \epsilon(2).\alpha.X + X) = \mu S : \{\alpha\} \cup S$. Clearly $\mu S : \{\alpha\} \cup S$ is $\{\alpha\}$.

3.2 Properties of Agents

For each $\sigma \in \mathcal{L}$, let $\xrightarrow{\sigma}$ be the least relation over $\mathcal{P}_R$, defined by the inference rules given in table 1. This yields a standard labelled transition system, $\langle \mathcal{P}_R, \mathcal{L}, \longrightarrow \rangle$. In the following, we state four properties of agents in terms of this system. They shall be considered as fundamental properties of real time behaviour. We will see that each of them has its corresponding equational laws.

Lemma 1 *(maximal progress) If $P \xrightarrow{\tau} P'$ for some P', then $P \xrightarrow{\epsilon(d)} P''$ for no d and P''.*

This is the most important property of our agents. Intuitively, it says that whenever an agent can perform a τ-action, it shall never wait. □

Lemma 2 *(time determinacy) Whenever $P \xrightarrow{\epsilon(d)} P_1$ and $P \xrightarrow{\epsilon(d)} P_2$ then $P_1 \equiv P_2$, where "$\equiv$" is the syntactical identity.*

When time goes, if an agent does nothing but idling, then it cannot reach different states. □

Lemma 3 *(time continuity) $P \xrightarrow{\epsilon(c+d)} P_2$ iff there exists P_1 such that $P \xrightarrow{\epsilon(c)} P_1$ and $P_1 \xrightarrow{\epsilon(d)} P_2$.*

This lemma says that if an agent proceeds from one instant to the other, it must reach all the intermediate instants between them. □

Lemma 4 *(persistency) If $P \xrightarrow{\epsilon(d)} P'$ and $P \xrightarrow{\alpha} Q$ for some Q, then $P' \xrightarrow{\alpha} Q'$ for some Q'.*

By idling, an agent shall not lose the ability of performing a controllable action that it is able to perform originally. □

Inaction	$\frac{}{\mathrm{NIL} \xrightarrow{\epsilon(d)} \mathrm{NIL}}$
Prefix	$\frac{}{\mu@t.P \xrightarrow{\mu} P[0/t]}$ $\qquad$ $\frac{}{\alpha@t.P \xrightarrow{\epsilon(d)} \alpha@t.P[t+d/t]}$
	$\frac{}{\epsilon(d+e).P \xrightarrow{\epsilon(d)} \epsilon(e).P}$ $\qquad$ $\frac{P \xrightarrow{\epsilon(d)} P'}{\epsilon(e).P \xrightarrow{\epsilon(d+e)} P'}$
	$\frac{P \xrightarrow{\sigma} P'}{\epsilon(0).P \xrightarrow{\sigma} P'}$
Summation	$\frac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'}$ $\qquad$ $\frac{Q \xrightarrow{\mu} Q'}{P+Q \xrightarrow{\mu} Q'}$
	$\frac{P \xrightarrow{\epsilon(d)} P' \qquad Q \xrightarrow{\epsilon(d)} Q'}{P+Q \xrightarrow{\epsilon(d)} P'+Q'}$
Composition	$\frac{P \xrightarrow{\mu} P'}{P\|Q \xrightarrow{\mu} P'\|Q}$ $\qquad$ $\frac{Q \xrightarrow{\mu} Q'}{P\|Q \xrightarrow{\mu} P\|Q'}$
	$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\bar{\alpha}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$
	$\frac{P \xrightarrow{\epsilon(d)} P' \qquad Q \xrightarrow{\epsilon(d)} Q'}{P\|Q \xrightarrow{\epsilon(d)} P'\|Q'}$ $\quad [Sort_d(P) \cap \overline{Sort_d(Q)} = \emptyset]$
Restriction	$\frac{P \xrightarrow{\sigma} P'}{P\backslash L \xrightarrow{\sigma} P'\backslash L}$ $\quad [\sigma, \bar{\sigma} \notin L]$
Relabelling	$\frac{P \xrightarrow{\sigma} P'}{P[S] \xrightarrow{S(\sigma)} P'[S]}$
Recursion	$\frac{E[recX : E/X] \xrightarrow{\sigma} P}{recX : E \xrightarrow{\sigma} P}$

where $\alpha \in \Lambda, \mu \in \mathcal{A}ct = \Lambda \cup \{\tau\}$ and $\sigma \in \mathcal{L} = \mathcal{A}ct \cup \delta_{\mathcal{R}+\infty}$.

Table 1: Operational Semantics of TCCS

4 Bisimulation and Equivalences

To complete the calculus, this section starts to establish equivalence relations between agents based on the well developed notion of bisimulation on labelled transition systems.

4.1 Strong Equivalence, $\sim$

As in CCS, we would like to identify two agents which cannot be distinguished by an observer (environment) interacting with them. Obviously, this equality depends on the *capability* of the observer. We first require that the observer is able to observe all kinds of actions and moreover, it can record the time delay between actions. So the set of observations is $\mathcal{L} = \Lambda \cup \{\tau\} \cup \delta_{\mathcal{R}^+0}$.

We define the notion of bisimulation on the system, $\langle \mathcal{P}_R, \mathcal{L}, \longrightarrow \rangle$ in the standard way [M89], called *strong bisimulation*. Let $\sim$ denote the largest strong bisimulation. As usual, $\sim$ is an equivalence relation and then by the properties of bisimulation, to prove the equivalence of two agents, we just need to construct a strong bisimulation containing the pair of agents. This gives rise to a simple proof technique for $\sim$.

For example, it is easy to check that $\mathcal{R}$ is a strong bisimulation, where $\mathcal{R} = \{(\alpha|\beta, \alpha.\beta + \beta.\alpha) | \alpha, \beta \in Act\} \cup \{(\mathrm{NIL}|P, P) | P \in \mathcal{P}_R\} \cup \{(P|\mathrm{NIL}, P) | P \in \mathcal{P}_R\}$. Therefore, $\alpha|\beta \sim \alpha.\beta + \beta.\alpha$. An interesting variant of this example is $\epsilon(1).\alpha|\epsilon(1).\beta \sim \epsilon(1).\alpha.\beta + \epsilon(1).\beta.\alpha$. Later, an expansion theorem will be developed, by which the above equivalences can be easily established.

In the standard way, $\sim$ can be extended to open expressions which may contain time and agent variables. Then, we have the following.

Theorem 1 *$\sim$ is preserved by all operators.*

Proof. We refer to [W91]. □

4.2 Weak Equivalence, $\approx$

In defining strong bisimulation equivalence, it is supposed that the observer is able to observe all actions, even non controllable ones and also capable of measuring time. Now let us deny the observer the ability of observing non controllable actions. We shall define a weak equivalence over agents in terms of controllable actions and time delays between them; whereas τ will be abstracted away from.

Let $\mathcal{E} = \Lambda \cup \Delta_\mathcal{T}$ be the set of observations, where Λ is the set of controllable actions and $\Delta_\mathcal{T} = \{\varepsilon(d) | d \geq 0\}$.

Definition 2

1. $P \stackrel{\alpha}{\Longrightarrow} Q$ *if* $P(\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^* Q$
2. $P \stackrel{\epsilon(d)}{\Longrightarrow} Q$ *if* $P(\stackrel{\tau}{\longrightarrow})^* \stackrel{\epsilon(d_1)}{\Longrightarrow} (\stackrel{\tau}{\longrightarrow})^* ... (\stackrel{\tau}{\longrightarrow})^* \stackrel{\epsilon(d_n)}{\Longrightarrow} (\stackrel{\tau}{\longrightarrow})^* Q$
   where $d = \sum_{i \leq n} d_i$.

Note that for each controllable action α, $P \stackrel{\alpha}{\Longrightarrow} Q$ is defined as usual in CCS. The following examples may help to understand $P \stackrel{\epsilon(d)}{\Longrightarrow} Q$. First, for every agent $P \in \mathcal{P}_R$, we have $P \stackrel{\epsilon(0)}{\Longrightarrow} P$. This is similar to $P \stackrel{\varepsilon}{\Longrightarrow} P$ in CCS. Since τ takes no time, $\tau.P \stackrel{\epsilon(0)}{\Longrightarrow} P$ and $\epsilon(2).\tau.\epsilon(3).P \stackrel{\epsilon(5)}{\Longrightarrow} P$.

Having introduced $\mathcal{E}$ and $\Longrightarrow$, we achieve a standard labelled transition system $\langle \mathcal{P}_R, \mathcal{E}, \Longrightarrow \rangle$. Naturally, we can establish the notion of bisimulation on this system in the standard way, which shall be called *weak bisimulation*. Let $\approx$ denote the largest weak bisimulation, called weak equivalence. As usual, $\approx$ is an equivalence relation and the weak equivalence between two agents can be proved by constructing a weak bisimulation containing the pair of agents. For example,

$\{(\tau.P, P)\} \cup \mathcal{I}d$ is a weak bisimulation and therefore $\tau.P \approx P$ as in CCS. This is a CCS τ-law. We will see that all standard CCS τ-laws are valid for $\approx$. Moreover, we shall have similar τ-laws for delay operator. For instance, $\epsilon(2).\tau.P \approx \epsilon(2).P$ and $\epsilon(2).\tau.\epsilon(3).P \approx \epsilon(5).P$.

Clearly, $\sim$ is a weak bisimulation and therefore $\approx$ is larger than $\sim$.

Proposition 1 *$P \sim Q$ implies $P \approx Q$.*

As usual, $\approx$ can be extended to open expressions. Like weak equivalence in CCS, $\approx$ is not a congruence either. However, we have the following.

Theorem 2 *$\approx$ is preserved by all operators except summation and recursion.*

Proof. For summation, the fact is well known. For recursion, we present the following counter example. It is obvious that $\tau.\alpha.(X + \beta.P) \approx \alpha.(X + \beta.P)$. But $recX : \tau.\alpha.(X + \beta.P) \not\approx recX : \alpha.(X + \beta.P)$, because the left hand side may refuse β after α and the right hand side is always able to offer β after α. For the other operators, we refer to [W91]. □

5 Equational Laws

The following presents equational laws in terms of bisimulation equivalences. We classify the laws into four groups. The first three groups are in terms of $\sim$ and the last group presents τ-laws in terms of $\approx$. Note that $\sim \subseteq \approx$ and therefore the laws for $\sim$ are also valid for $\approx$.

5.1 Basic Laws

This group includes the basic laws from CCS and their timed variants.

Proposition 2

$$
\begin{array}{ll}
E + F \sim F + E & E|F \sim F|E \\
(E + F) + G \sim E + (F + G) & (E|F)|G \sim E + (F + G) \\
E + E \sim E & E|NIL \sim E \\
E + NIL \sim E &
\end{array}
$$

$$
\begin{array}{ll}
(E + F)\backslash L \sim F\backslash L + E\backslash L & (E + F)[S] \sim F[S] + E[S] \\
(\mu@t.E)\backslash L \sim NIL \quad (\mu \in L \cup \overline{L}) & (E|F)[S] \sim F[S]|E[S] \\
(\mu@t.E)\backslash L \sim \mu@t.(E\backslash L) \ (\mu \notin L \cup \overline{L}) & (\mu@t.E)[S] \sim S(\mu)@t.(E[S]) \\
(\epsilon(d).E)\backslash L \sim \epsilon(d).(E\backslash L) & (\epsilon(d).E)[S] \sim \epsilon(d).(E[S]) \\
NIL\backslash L \sim NIL & NIL[S] \sim NIL
\end{array}
$$

$$recX : E \sim E[recX : E/X]$$

As claimed earlier, we do not distinguish $\mu@t.E$ from $\mu@u.E[u/t]$. This is justified by the following law.

Proposition 3 *(α-conversion) $\mu@t.E \sim \mu@u.E[u/t]$*

It is worth to point out that the expansion theorem of CCS is also valid for $\sim$. It is just a simple case of the expansion theorem given later.

5.2 Real Time Laws

The laws above do not concern explicitly with timing. The following shall deal with the real time properties of agents. Each of the four basic properties of agents formalized in section 3 has its corresponding laws.

Proposition 4

1. *(maximal progress)*

$$\tau@t.E \sim \tau.E[0/t]$$
$$\tau.E + \epsilon(d).F \sim \tau.E$$

2. *(time determinacy)*

$$\epsilon(d).(E + F) \sim \epsilon(d).E + \epsilon(d).F$$
$$\epsilon(d).(E|F) \sim \epsilon(d).E|\epsilon(d).F$$

3. *(time continuity)*

$$\epsilon(c + d).E \sim \epsilon(c).\epsilon(d).E$$
$$\epsilon(0).E \sim E$$
$$\epsilon(d).NIL \sim NIL$$

4. *(persistency)*

$$\alpha@t.E + \epsilon(d).\alpha@t.E[t + d/t] \sim \alpha@t.E$$

5.3 Expansion Theorem

We first, consider a particular case dealing with agents which have no delay before their first actions and no time variable associated with their first actions either.

Proposition 5 *Let* $E \equiv \sum_{i\in I} \mu_i.E_i$ *and* $F \equiv \sum_{j\in J} \nu_j.F_j$. *Then*

$$E|F \sim \sum_{\mu_i=\overline{\nu_j}\in\Lambda} \tau.(E_i|F_j) + \sum_{i\in I} \mu_i.(E_i|F) + \sum_{j\in J} \nu_j.(F_j|E)$$

This law is essentially the expansion theorem of CCS [M89]. For instance, we have $\alpha|\beta \sim \alpha.\beta + \beta.\alpha$. Now we turn to a more general case that allows delays before the first actions of agents, but no time variables. We look at an example $\epsilon(2).\alpha|\beta$ to motivate why we need to introduce time variables to achieve an expansion. We wish to find a regular agent equivalent to the composite one.

Observe that after α, $\epsilon(2).\alpha|\beta$ is able to perform β immediately. But after β, it may take some time for $\epsilon(2).\alpha|\beta$ to enable α. For instance, if β is performed by time $t = 2$, then it can perform α immediately. But if β is performed by time $t = 1.5$, it will take 0.5 units of time before it can perform α. This means that the following agent after β depends on the time when β is performed. Therefore, we need to associate a time variable to action β in the expansion bellow.

$$\epsilon(2).\alpha|\beta \sim \epsilon(2).\alpha.\beta + \beta@t.\epsilon(2 \dot{-} t).\alpha$$

This example is a simple application of the following law.

Proposition 6 *Let* $E \equiv \sum_{i\in I} \epsilon(c_i).\mu_i.E_i$ *and* $F \equiv \sum_{j\in J} \epsilon(d_j).\nu_j.F_j$. *Then*

$$E|F \sim \sum_{\mu_i=\nu_j\in\Lambda} \epsilon(\max(c_i,d_j)).\tau.(E_i|F_j) + \sum_{i\in I}\epsilon(c_i).\mu_i@x_i.(E_i|F') + \sum_{j\in J}\epsilon(d_j).\nu_j@y_j.(E'|F_j)$$

where

- $F' \equiv \sum_{j\in J}\epsilon(d_j \dot{-} c_i \dot{-} x_i).\nu_j.F_j$
- $E' \equiv \sum_{i\in I}\epsilon(c_i \dot{-} d_j \dot{-} y_j).\mu_i.E_i$

For instance, $\epsilon(3).\alpha|\epsilon(1).\beta \sim \epsilon(3).\alpha@t.\epsilon(1\dot{-}3\dot{-}t).\beta + \epsilon(1).\beta@t.\epsilon(3\dot{-}1\dot{-}t).\alpha$. By the properties of $\dot{-}$, the right hand side can be further simplified, $\epsilon(3).\alpha|\epsilon(1).\beta \sim \epsilon(3).\alpha.\beta + \epsilon(1).\beta@t.\epsilon(2\dot{-}t).\alpha$. Note that the previous law is a simple case of this law when all $c_i, d_i = 0$.

Finally, we take away the restrictions imposed on E and F above and turn to the general case.

Proposition 7 *(Expansion Theorem)*
Let $E \equiv \sum_{i\in I}\epsilon(c_i).\mu_i@x_i.E_i$ *and* $F \equiv \sum_{j\in J}\epsilon(d_j).\nu_j@y_j.F_j$. *Then*

$$E|F \sim \sum_{\mu_i=\nu_j\in\Lambda} \epsilon(\max(c_i,d_j)).\tau.(E_i[d_j \dot{-} c_i/x_i]|F_j[c_i \dot{-} d_j/y_j]) + \sum_{i\in I}\epsilon(c_i).\mu_i@x_i.(E_i|F') + \sum_{j\in J}\epsilon(d_j).\nu_j@y_j.(E'|F_j)$$

where

- $F' \equiv \sum_{j\in J}\epsilon(d_j \dot{-} c_i \dot{-} x_i).\nu_j@y_j.\{F_j[y_j + ((c_i + x_i) \dot{-} d_j)/y_j]\}$
- $E' \equiv \sum_{i\in I}\epsilon(c_i \dot{-} d_j \dot{-} y_j).\mu_i@x_i.\{E_i[x_i + ((d_j + y_j) \dot{-} c_i)/x_i]\}$

Proof. We refer to [W91]. □

For instance, $\epsilon(2).\alpha|\beta@t.\epsilon(t).E \sim \epsilon(2).\alpha@u.\beta@t.\epsilon(t+2+u).(E[t+u+2/t]) + \beta@t.(\epsilon(2\dot{-}t).\alpha|E)$.

Note that the expansion theorem includes the first two expansion laws and moreover, the expansion is static in the sense that it ignores the possibility that some of the summands on the right-hand side may be eliminated directly. However the sum may be further reduced by the maximal progress law.

5.4 τ-Laws

In terms of $\sim$, we have established a number of laws. They are also valid for $\approx$ because $\approx$ is larger than $\sim$. Moreover, we have the following standard CCS τ-laws which hold for $\approx$.

Proposition 8

$\tau.E \approx E$
$\tau.E + E \approx \tau.E$
$\mu.\tau.E \approx \mu.E$
$\mu.(E + \tau.F) \approx \mu.(E + \tau.F) + \mu.F$

For the last two laws, we have the following generalization to timed action prefix.

Proposition 9

$\mu@t.\tau.E \approx \mu@t.E$
$\mu@t.(E + \tau.F) \approx \mu@t.(E + \tau.F) + \mu@t.F$

For delay operator, we have analogous laws.

Proposition 10

$\epsilon(d).\tau.E \approx \epsilon(d).E$
$\epsilon(d).(E + \tau.F) \approx \epsilon(d).(E + \tau.F) + \epsilon(d).F$

6 Related Work

An earlier work on the subject has been reported in [W90a]. Unfortunately, the expansion theorem described in [W90a] does not work. This leads to the present work. The main new ideas in this paper include the timed action prefix $\mu@t.P$ to deal with time dependence, which, together with the assumption that actions are instantaneous, gives rise to a natural extension of the expansion theorem of CCS. Also, the *maximal progress* assumption which has appeared in [RH89], but not in the context of process calculi, and *time determinacy* assumption are formalized in an operational framework.

Recently, various timed models have been proposed [HR90, MT90, OF90, RR86, S90, W90a, J91]. For a short survey, we refer the reader to [OF90, S90, W91]. Most of these models have in some way introduced the maximal progress and time determinacy assumptions except [MT90] which avoids the maximal progress assumption. However, none of these models has used time variables in their processes. More recently, another timed CCS has been reported in [CAM90], which combines the delay construct and timed action prefix together to one prefix operator $\alpha(t)|_c^e.P$. Intuitively, the agent offers α within the interval $[c, e]$ and then become $P[d/t]$, where $d \in [c, e]$ is the delay before α is performed. In fact, $\alpha(t)|_c^\infty.P$ has the same meaning as $\epsilon(c).\alpha@t.P[t+c/t]$ and moreover, the calculus is also based on the maximal progress and time determinacy assumptions.

Acknowledgements: I am grateful for discussions with Uno Holmer, Alan Jeffrey, Kim Larsen, Chen Liang, Steve Schneider and Bjorn von Sydow, and to the anonymous referee for their comments and suggestions. In particular, thanks my advisor Bjorn von Sydow for his encouragement.

References

[CAM90] L. Chen and S. Anderson and F. Moller, *A Timed Calculus of Communicating Systems*, LFCS report-90-127, Edinburgh University, December 1990.

[HR90] M. Hennessy and T. Regan, *A Temporal Process Algebra*, Report No 2/90, University of Sussex, April 1990.

[J91] A. Jeffrey, *Linear Time Process Algebra*, (submitted to CAV91), Dept. of Computer Sciences, Chalmers University, Sweden, March 1991.

[M83] R. Milner, *Calculi for Synchrony and Asynchrony*, TCS, Vol 25, 1983.

[M89] R. Milner, *Communication and Concurrency*, Prentice Hall, 1988.

[MT90] F. Moller and C. Tofts, *A Temperal Calculus of Communicating Systems*, LNCS, No. 458, 1990.

[OF90] Y. Ortega-Mallen and D. de Frutos-Escrig, *Timed Observations: a semantic model for real time concurrency*, In M. Broy and C.B. Jones, editors, TC2-Working Conference on Programming Concepts and Methods, 1990.

[RH89] W.P. de Roever and J.J.M. Hooman, *Design and Verification of Real-time Distributed Computing: an Introduction to Compositional Methods*, Proceeding of the 9th IFIP WG 6.1 International Symposium on Protocol Specification, Testing and Verification, 1989.

[RR86] G.M. Reed and A.W. Roscoe, *A Timed Model for Communicating Sequential Processes*, LNCS, No. 226, 1986.

[S90] J. Sifakis et al. *The Algebra of Timed Processes ATP: Theory and Application*, Laboratoire de Genie Informatique, IMAG-Campus, B.P.53X, 38041 Grenoble Cedex, France, December 1990.

[W90a] Y. Wang, *Real Time Behaviour of Asynchronous Agents*, LNCS, No. 458, 1990.

[W90b] Yi Wang, *CCS + Time = an Interleaving Model for Real Time Systems*, Nordic Workshop on Program Correctness (internal report), Aalborg University, Denmark, October 1990.

[W91] Y. Wang, *A Calculus of Real Time Systems*, Ph.D. thesis (in preparation), Dept. of Computer Sciences, Chalmers University, Sweden, 1991.