TCCT: Timed Calculus with Controllable Timers

A. Itou[1,*] Y. Hayashi[2] R. Nakajima[1]
M. Tanabe[3]
[1]Research Institute for Mathematical Sciences, Kyoto University [†]
[2]Omron Corp.
[3]Ube National College of Technology

April 30, 2002

Abstract

A timed process calculus TCCT is introduced, to model computing mechanisms which deal with complicated time critical situations, where various events occur and each event occurrence can be affected by when and what events occurred so far. We often encounter this kind of phenomena, for which it seems significant to introduce a formal computing model.

Considering the nature of the model, it is crucial at which time points and with what time intervals event occur. In order to deal with such temporal information, it is necessary to nicely control multiple time constraints on a single process. Therefore we define the process calculus TCCT so that a multiple number of labelled timers can be assigned to a single process and separately release each timer by referring to its label.

In most conventional process calculi, it is assumed that everything in question can be described as processes, no matter how large the system is. There the main purpose of discussion is usually to analyse the ways of communication between such processes. In the real world, timed systems are very large and complicated, and, so are the processes which model them. In this calculus, however, such systems are considered only as an event generator, which is useful to simplify the situation and in the result to get simpler processes. This perspective is also a contribution of the calculus.

In addition, in this paper we show that the practical feasibility of the calculus by actually using TCCT to implement a certain recognising devices, besides discussing the calculus merely as a formal computing model.

Finally with regard to a mathematical property of the calculus, it is demonstrated that an equality based on the conventional bisimulation is preserved by all contexts for a reasonable class of processes.

1 Introduction

We introduce a timed process calculus called TCCT (Timed Calculus with Controllable Timers) which are computing mechanisms to deal with complicated time critical situations, where various events occur and each event occurrence affects one another. In this model, it is crucial at which time points and with what time intervals event occur, because occurrence of each event can be affected by when and what event occurred so far. To cope with such

*Corresponding author. Email: akira@kurims.kyoto-u.ac.jp
†Kyoto, 606-8502 Japan

circumstances, the calculus presented in this paper is designed to deal with time constraints of event occurrences exquisitely. This model expresses quite a general phenomenon, for which it seems significant to introduce a formal computing model.

By and large, in most conventional process calculi, it is assumed that everything in question can be described as processes, no matter how large the system of an object is. There the main purpose of discussion is usually to analyse the ways of communication between such processes. Usually the reactive/real-time systems to be modelled are very large and complicated and so the processes tend to be very large and complicated. Therefore we consider the timed systems only as an event generator, which is useful to simplify the situation and in the result to get simpler processes. This perspective is also a contribution of TCCT.

For the reason above, with a TCCT it is possible to nicely control multiple time constraints on a single process, which is enabled by a labelling mechanism to timers. We can assign labelled timers to processes and separately release each timer by referring to its label. A TCCT process waits more than one events which are output from the environment (or the event generator) with a time limit for the arrival. If an event arrives within its time limit, it performs some particular task corresponding to the event. Otherwise it goes to some exceptional task. Also, all processes that are waiting for an event, not a non-deterministically selected one, react parallelly to the event simultaneously. For simplicity, unlike conventional timed process calculi, we do not include in TCCT the mechanism of inter-process communication, though it is not an essential restriction.

In TCCT, passage of time and execution of actions are separately treated. There, time passage is modelled by a time progress transition, and execution of actions is supposed to be instantaneous, which results in generating processes diverging without time progress. These unrealistic processes are not particular to TCCT, but are common with most of the other timed process calculi that deal with time passage in the same way as TCCT. In order to cope with the problem, we restrict processes syntactically to exclude such unrealistic ones from our consideration.

In addition, the syntax of TCCT, which introduces the mechanism of releasing a labelled timer by referring to its label, generates uninterpretable processes, such as a process which attempts to release a timer in spite of that the timer is not at all set on it. To get around, we see that we treat only the class of reasonably interpretable processes, which we define as 'admissible processes'. For admissible processes, we show that the equivalence relation, which is given with the conventional notion of bisimulation, is preserved by all contexts, while it is not the case for all processes including such pathological ones.

In this paper, in addition to discussing TCCT as a formal computing model, we show that the practical feasibility of the calculus with a concrete example of using a TCCT process to implement a 'timed event pattern matching recognizer' [3].

One way of getting around the difficulty with time-critical systems, which we mentioned at the beginning of this section, is merely to watch and find out when and how the system changes into some specific states, instead of continuously watching the global state space. By regarding reactive and real-time systems as timed event generators, instead of attempting to model behaviour of such systems in terms of state transitions, it is often convenient to find some of their crucial characteristics from the generated sequence of events together with their time points. This is the paradigm called timed event pattern matching, for which TCCT is found to be useful to write recognizing devices though, in [4], a state-base machine with a dynamic number of states was used, instead of TCCT.

In Section 2 we describe the syntax and the operational semantics of TCCT. To show what the calculus can do, as an example we actually implement a recognising device, using a TCCT process in Section 2.4. An equivalence relation of TCCT based on bisimulation is given and proved to be congruent in Section 3. In the final section we discuss the problem

of treating the notion of time in the calculus.

Related Work

Several versions of timed process calculi are presented [9], such as Temporal CCS [8], Timed CCS [13], TPL [5], ATP [10], TCSP [12], and ACP$_\rho$ [1].

Timed process calculi with time-out operators are presented in ATD$_D$ [11], ATP, TPCCS [2], and TCSP (not exclusive). TCSP is equipped with a mechanism to cancel timers, though it appears that mechanism of timer labelling is not included in any other timed calculi than TCCT.

For the timed event pattern matching, we refer to [3, 4].

2 Timed Calculus with Controllable Time-Out

Now we introduce the calculus TCCT (Timed Calculus with Controllable Time-Out) that is a version of timed process calculi.

TCCT has the following features, compared with other timed calculi.

- Reacting to events: Processes wait and react to events and emit signals to the environment. If more than a single processes wait for a same event to come, all of them, not a non-deterministically selected one, react simultaneously when the event arrives.
- Labelled timers: We can assign the labelled timers to a process. There is an operation to release each timer by referring to its label, so that we can control multiple time constraints exquisitely.

2.1 Syntax

We presuppose the time domain $Time$, which is the set of non-negative real numbers.

Let PC be the set of *process constants*, and TN be the infinite set of *timer-names* , EV_{IN} be the set of events input from the environment, and EV_{OUT} be the set of events output to the environment.

Usually we use A, B, C over PC, S, T, U over TN , $a, b, c, ...$ over EV_{IN}, $\alpha, \beta, \gamma, ...$ over EV_{OUT}, and s, t over $Time$

Definition 1 *The set $\mathcal{P}$ of TCCT processes is defined as follows.*

$$P \quad ::= \quad 0 \mid \alpha \mid a.P \mid \mathit{off}\langle T\rangle.P \mid P^{\{T:t\}}P \mid P \parallel P \mid P + P \mid A(T_1, \ldots, T_n)$$

In $P^{\{T:t\}}Q$ we say that the timer-name T occurring in P is bound and Q is the exception process. We write $\mathbf{ft}(P)$ for the set of all timer-names occurring free in P.

We also assume that for each process constant A there exists a defining equation of the form $A(T_1, \ldots, T_n) \stackrel{def}{=} P_A$, where $T_1, \ldots, T_n$ are a list of $\mathbf{ft}(P_A)$ (omitted when clear from the context).

As usual, 0 is often omitted after an event, e.g. $a.0$ as a. If an exception process of a timer is 0 (e.g. $P^{\{T:t\}}0$), we may omit 0 ($P^{\{T:t\}}$).

We informally read the above expressions (which are reasonably interpretable as we mention below) as follows.

- $a.P$ waits for the event a from the environment, and evolves into P on receiving a.

- α instantly outputs the event α to the environment.
- $P^{\{T:t\}}Q$ means that the timer T is set on the process P, and then turns to Q after t time units.
- $\mathit{off}\langle T\rangle$ releases the timer T, namely $(\mathit{off}\langle T\rangle.P)^{\{T:t\}}Q$ changes into P.
- The others are much the same as in CCS.

It might be noticed that the syntax rule above yields expressions to which reasonable interpretation cannot be given, such as $\mathit{off}\langle T\rangle.a$ and $(\mathit{off}\langle T\rangle.\mathit{off}\langle T\rangle.a)^{\{T:t\}}(b.c)$. An alternative complicated syntax rule could eliminate them, but we prefer simplicity for syntax, and along with semantics consideration we will only deal with reasonably interpretable expressions. We will discuss more on this issue later in this section.

Congruence on time constraints

We impose the following congruence relation on TCCT.

Definition 2 *The congruence relation, $\equiv$ over $\mathcal{P}$ is defined as follows.*

TC-rename	$P^{\{T:t\}}Q$	$\equiv$	$(P[T'/T])^{\{T':t\}}Q$
TC-comm.time	$(P^{\{T:t\}}Q)^{\{S:s\}}R$	$\equiv$	$(P^{\{S:s\}}R)^{\{T:t\}}Q$
TC-comm.par	$P \parallel Q$	$\equiv$	$Q \parallel P$
TC-comm.sum	$P + Q$	$\equiv$	$Q + P$
TC-distri.par	$(P \parallel Q)^{\{T:t\}}R$	$\equiv$	$P^{\{T:t\}}R \parallel Q^{\{T:t\}}R$
TC-distri.sum	$(P + Q)^{\{T:t\}}R$	$\equiv$	$P^{\{T:t\}}R + Q^{\{T:t\}}R$
TC-rec	$A(T_1, \ldots, T_n)$	$\equiv$	P_A

where $T' \notin \mathbf{ft}(P)$, $T \neq S$, $A(T_1, \ldots, T_n) \stackrel{def}{=} P_A$, and $P[T'/T]$ is derived from P by replacing T occurring free in P, with T'.

TC-distri.par is necessary to control a time constraint separately over some different processes.

2.2 Operational Semantics

In giving meaning to TCCT, we use the notion of a labelled transition system. The transition rules of TCCT are given in Table 1.

Let us take a look at some rules in detail.

Time progress

These rules indicate how time passes to TCCT processes and when a timer starts working.

Weakly guarded [7] timers (such as the timer $\{T : t\}$ in $a.(P^{\{T:t\}}Q)$) do not get affected by time progress. So $a.((P)^{\{T:t\}}Q) \xrightarrow{s} a.((P)^{\{T:t\}}Q)$ but $a.((P)^{\{T:t\}}Q) \not\xrightarrow{s} a.((P)^{\{T:t-s\}}Q)$.

It is noticed that we do not have any time-progress transition of processes in the form of α or $\mathit{off}\langle T\rangle.P$, so that such processes output the event or release the timer without any delay. That is why the calculus satisfies a sort of what is called the *maximal progress assumption*: when a process is able to output events or to release some timers, it does not wait to do unnecessarily.

Time progress

$$\frac{}{0 \xrightarrow{t} 0} \qquad \frac{}{a.P \xrightarrow{t} a.P}$$

$$\frac{P \xrightarrow{t'} P'}{P^{\{T:t\}}Q \xrightarrow{t'} P'^{\{T:t-t'\}}Q}\ (t \geq t') \qquad \frac{P \xrightarrow{t} P' \quad Q \xrightarrow{t} Q'}{P \parallel Q \xrightarrow{t} P' \parallel Q'}$$

$$\frac{P \xrightarrow{t} P' \quad Q \xrightarrow{t} Q'}{P + Q \xrightarrow{t} P' + Q'} \qquad \frac{P \equiv P' \quad P' \xrightarrow{a} Q' \quad Q' \equiv Q}{P \xrightarrow{a} Q}$$

Event-in

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{P^{\{T:t\}}Q \xrightarrow{a} P'^{\{T:t\}}Q}\ (t > 0)$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P' \parallel Q'} \qquad \frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q}\ (a \notin wait(Q))$$

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \frac{P \equiv P' \quad P' \xrightarrow{a} Q' \quad Q' \equiv Q}{P \xrightarrow{a} Q}$$

Event-out

$$\frac{}{\alpha \xrightarrow{\alpha} 0} \qquad \frac{P \xrightarrow{\alpha} P'}{P^{\{T:t\}} \xrightarrow{\alpha} P'^{\{T:t\}}}\ (t > 0)$$

$$\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P' + Q}$$

$$\frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q}$$

Timer

$$\frac{}{P^{\{T:0\}}Q \xrightarrow{\tau} Q} \qquad \frac{}{(\mathit{off}\langle T\rangle.P)^{\{T:t\}}Q \xrightarrow{\tau} P}\ (t > 0)$$

$$\frac{P \xrightarrow{\tau} P'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q} \qquad \frac{P \equiv P' \quad P' \xrightarrow{\tau} Q' \quad Q' \equiv Q}{P \xrightarrow{\tau} Q}$$

$$\frac{P \xrightarrow{\tau} P'}{P + Q \xrightarrow{\tau} P'} \qquad \frac{P \xrightarrow{\tau} P'}{P^{\{T:t\}} \xrightarrow{\tau} P'^{\{T:t\}}}\ (T \in \mathbf{ft}(P) \vee T \notin \mathbf{ft}(P'))$$

Table 1: Transtion Rules of TCCT

$$
\begin{array}{lcl}
wait(0) = wait(\alpha.P) = wait(\mathit{off}\langle T\rangle.P) & = & \emptyset \\
wait(a.P) & = & \{a\} \\
wait(P \parallel Q) = wait(P + Q) & = & wait(P) \cup wait(Q) \\
wait(P^{\{T:t\}}) & = & wait(P) \\
wait(A(T_1, \ldots, T_n)) & = & wait(P_A)
\end{array}
$$

$(where\ A(T_1, \ldots, T_n) \stackrel{def}{=} P_A)$

Table 2: Definition of *wait*

Event-in

All processes respond only to the environment, but they cannot communicate with each other. Unlike CCS, therefore, all the processes in parallel react to an external event simultaneously. Namely, we see that on receiving a, $a.P \parallel a.Q$ has the only one possible transition: $a.P \parallel a.Q \stackrel{a}{\rightarrow} P \parallel Q$.

The function *wait* is of $\mathcal{P} \rightarrow 2^{EV}$ and $wait(P)$ includes all the events that P is waiting for. We define the function in Table 2, on the assumption that process constants are weakly guarded, which is defined below. But for the assumption, it would be a rather harder work.

Event-out

These are almost the same as in the conventional ones.

Timer

When the deadline comes to a timer of a process, the working process is discarded and replaced by the exceptional process.

As mentioned before, the time progress rules (i.e. $\stackrel{t}{\rightarrow}$) cannot be applied to a process prefixed by *off* (e.g. $(\mathit{off}\langle T\rangle.P)^{\{T:t\}}$), before releasing the timers. The timers, therefore, are released with no time delay.

In order to achieve flexible control of multiple constraints, it is better to control the time constraint $\{T : t\}$ in a process $(P \parallel Q)^{\{T:t\}}$ independently , i.e., release of the timer by P should not affect the time constraint of Q. So we introduce the congruence relation $\equiv$ so as to equate $(P \parallel Q)^{\{T:t\}}$ with $P^{\{T:t\}} \parallel Q^{\{T:t\}}$, and we design the transition system so that parallel processes having a common timer (e.g. $(P \parallel Q)^{\{T:t\}}$) cannot evolve unless the timer is distributed to each process (e.g. $P^{\{T:t\}} \parallel Q^{\{T:t\}}$).

In the following example, for events $a_1, a_2, \ldots$ and time durations $t_1, t_2, \ldots$, $\langle t_1\rangle \cdot a_1 \cdot \langle t_2\rangle \cdot a_2 \cdot \ldots$ stands for a timed event sequence with obvious interpretation.

Example 1

Here is a simple example to illustrate how a TCCT process evolves. Given the process $a.(b.\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:3\}}(d.0)$, and suppose the following sequence is input into the process $\langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot b \cdot \langle 2\rangle \cdot d \cdot \cdots$. Then we have the transitions as follows.

$$
\begin{array}{lll}
 & a.(b.\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:3\}}(d.0) & \\
\xrightarrow{1} & a.(b.\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:3\}}(d.0) & \text{(1 sec. progress)} \\
\xrightarrow{a} & (b.\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:3\}}(d.0) & \text{(Event } a \text{ occurs.)} \\
\xrightarrow{2} & (b.\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:1\}}(d.0) & \text{(2 sec. progress)} \\
\xrightarrow{b} & (\mathit{off}\langle T\rangle.\alpha \parallel c.Q)^{\{T:1\}}(d.0) & \text{(Event } b \text{ occurs.)} \\
\equiv & (\mathit{off}\langle T\rangle.\alpha)^{\{T:1\}}(d.0) \parallel (c.Q)^{\{T:1\}}(d.0) & (T \text{ is distributed (\textbf{TC-distri.par}).)} \\
\xrightarrow{\tau} & \alpha \parallel (c.Q)^{\{T:1\}}(d.0) & \text{(The timer } T \text{ of the left is released.)} \\
\xrightarrow{\alpha} & 0 \parallel (c.Q)^{\{T:1\}}(d.0) & \text{(Event } \alpha \text{ occurs.)} \\
\xrightarrow{1} & 0 \parallel (c.Q)^{\{T:0\}}(d.0) & \text{(1 sec. progress)} \\
\xrightarrow{\tau} & 0 \parallel d.0 & \text{(Time-out. Process } c.Q \text{ is killed.)} \\
\xrightarrow{d} & 0 \parallel 0 & \text{(Event } d \text{ occurs.)} \\
\cdots & &
\end{array}
$$

Therefore we get the output sequence: $\langle 3\rangle \cdot \alpha \cdot \langle 1\rangle \cdot \cdots$.

Example 2

With the mechanism of the time-out exception, we can get the following process Cl ($\stackrel{def}{=} 0^{\{T:t\}}(\alpha \parallel Cl)$) which outputs a signal α at intervals of t time units.

$$Cl \xrightarrow{t} 0^{\{T:0\}}(\alpha \parallel Cl) \xrightarrow{\tau} \alpha \parallel Cl \xrightarrow{\alpha} Cl \xrightarrow{t} 0^{\{T:0\}}(\alpha \parallel Cl) \xrightarrow{\tau} \alpha \parallel Cl \xrightarrow{\alpha} Cl \xrightarrow{t} \cdots$$

The output sequence of Cl is $\langle t\rangle \cdot \alpha \cdot \langle t\rangle \cdot \alpha \cdot \cdots$.

2.3 Admissible Processes

As already mentioned, the syntax rule unfortunately generates processes with no reasonable interpretation.

First, as in most process calculi, non-*weakly-guarded* process constants are troublesome because their behaviours are indefinite. *Weakly guarded* process constants are defined in TCCT as follows:

Definition 3 *An process constant A is weakly guarded in P if each occurrence of A is within some subprocess of P which is prefixed some input event.*

For instance, in $a.(b \parallel A)$ the process constant A is weakly guarded, while in $(\mathit{off}\langle T\rangle.B)^{\{T:t\}}$ and $\mathit{off}\langle T\rangle.(\alpha \parallel B)$ B is not weakly guarded because B is in a form of $\mathit{off}\langle T\rangle.P$ but not in a subprocess prefixed some input event. Note that we discuss in Section 4 the reason of 'prefixed some input event', not 'prefixed some input event or some $\mathit{off}\langle T\rangle$' in the definition above.

Following the conventional, we only treat process constants that are weakly guarded in their defining equations.

Furthermore, due to labelled timers and the bounded operators, there are another kind of syntactically odd processes. We must take a good care of those which contain occurrence of free timer-names (such as, $\mathit{off}\langle T\rangle.P, (\mathit{off}\langle T\rangle.P)^{\{S:s\}}$) as well as those which are reducible to such processes (such as, $(\mathit{off}\langle T\rangle.\mathit{off}\langle S\rangle.P)^{\{T:t\}}$, $(\mathit{off}\langle T\rangle.\mathit{off}\langle T\rangle.P)^{\{T:t\}}$). For instance, as we see in the transition rules above, such a process cannot evolve at all, or eventually turns into a process which cannot evolve at all.

There are many other ways to cope with this issue. As the result of our consideration, we see that we treat only the following *admissible* processes. Almost all the processes we use in application, are admissible. Furthermore, we have technical advantages to treating only the admissible process; we show later in Section 3 that *the "equality" of the admissible processes is preserved by all contexts, while it is not the case for unadmissible ones.*

Definition 4 *P is an admissible process iff for any P' such that $P \xrightarrow{*} P'$, P' contains no occurrence of free timer-names, where $\xrightarrow{*}$ is the transitive reflexive closure of $\xrightarrow{\ell}$.*

Admissible processes can be specified syntactically as follows.

Proposition 1 *P is an admissible processes if P is in $\{P \mid Adm(P, \emptyset, \emptyset)\}$, where*

$$
\begin{array}{lcl}
Adm(0, \mathcal{TN}, \mathcal{PC}) & = & \mathit{True} \\
Adm(a.P, \mathcal{TN}, \mathcal{PC}) & = & Adm(P, \mathcal{TN}, \mathcal{PC}) \\
Adm(P_1 \parallel P_2, \mathcal{TN}, \mathcal{PC}) & = & Adm(P_1, \mathcal{TN}, \mathcal{PC}) \wedge Adm(P_2, \mathcal{TN}, \mathcal{PC}) \\
Adm(P_1 + P_2, \mathcal{TN}, \mathcal{PC}) & = & Adm(P_1, \mathcal{TN}, \mathcal{PC}) \wedge Adm(P_2, \mathcal{TN}, \mathcal{PC}) \\
Adm(\mathit{off}\langle T\rangle.P, \mathcal{TN}, \mathcal{PC}) & = & Adm(P, \mathcal{TN} \backslash \{T\}, \mathcal{PC}) \quad (\textbf{if} \quad T \in \mathcal{TN}) \\
 & = & \mathit{False} \quad (\textbf{if} \quad T \notin \mathcal{TN}) \\
Adm(P^{\{T:t\}}Q, \mathcal{TN}, \mathcal{PC}) & = & Adm(P, \mathcal{TN} \cup \{T\}, \mathcal{PC}) \wedge Adm(Q, \mathcal{TN}, \mathcal{PC}) \\
Adm(A, \mathcal{TN}, \mathcal{PC}) & = & Adm(P_A, \mathcal{TN}, \mathcal{PC} \cup \{A : \mathcal{TN}\}) \\
 & & (\textbf{if} \quad \{A : \mathcal{L}\} \notin \mathcal{PC}) \\
 & = & (\mathcal{TN} \supseteq \mathcal{L}) \quad (\textbf{if} \quad \{A : \mathcal{L}\} \in \mathcal{PC})
\end{array}
$$

Here $\mathcal{TN}$ and $\mathcal{L}$ are sets of timer-names, and $\mathcal{PC}$ is a set whose elements are sets of timer-names. Each element of $\mathcal{PC}$ is labelled by process constants.

Proof
It is straightforward by the definition of *Adm*.

2.4 Example: Exhaustive Timed Event Pattern Matching Recognizer

We present an example to show how to use the calculus. It is the *exhaustive matching recognizer* for EPG (Event Pattern Graph) patterns [3], which can be implemented using TCCT processes. EPG patterns are a certain class of extended directed graphs with which we can represent timed patterns of events. With the exhaustive matching, a signal is output in real time whenever a matching for a given pattern is succeeded.

The exhaustive matching recognizer for the *epg* (Figure 1) receives events $a, b, c, d, e, \ldots$ which occur in the environment and it finds out all possible matchings with respect to the pattern *epg*.

Intuitively, the parts of the *epg* read in the following manner:

- The edge $\textcircled{a}_{n_1} \xrightarrow{5} \textcircled{c}_{n_3}$ represents the pattern that *a occurs and then c occurs within 5 seconds.*
- The edge $\textcircled{b}_{n_2} \xrightarrow[\{d\}]{} \textcircled{c}_{n_3}$ represents the pattern that *a, and then c occur without occurrence of d in between.*

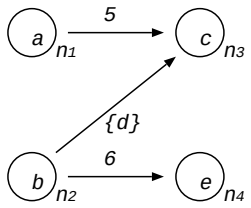

Figure 1: *epg*

For example, suppose that the following event sequence occurs:

$$\langle 2\rangle \cdot b \cdot \langle 1\rangle \cdot c \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot e \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot e \cdot \langle 1\rangle \cdot b \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot c \cdot \langle 1\rangle \cdot d \cdot \langle 1\rangle \cdot e \cdot \cdots,$$

which matches the *epg* in two ways:

$$\langle 2\rangle \cdot \underline{b} \cdot \langle 1\rangle \cdot c \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot \underline{e} \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot e \cdot \langle 1\rangle \cdot b \cdot \langle 1\rangle \cdot \underline{a} \cdot \langle 2\rangle \cdot \underline{c} \cdot \langle 1\rangle \cdot d \cdot \langle 1\rangle \cdot e \cdot \cdots$$

and

$$\langle 2\rangle \cdot b \cdot \langle 1\rangle \cdot c \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot e \cdot \langle 1\rangle \cdot a \cdot \langle 2\rangle \cdot e \cdot \langle 1\rangle \cdot \underline{b} \cdot \langle 1\rangle \cdot \underline{a} \cdot \langle 2\rangle \cdot \underline{c} \cdot \langle 1\rangle \cdot d \cdot \langle 1\rangle \cdot \underline{e} \cdot \cdots,$$

where the event occurrences, which are detected in each matching, are underlined.

Here we are going to give a concrete design of the recognizer for the *epg*.

In order to implement the recognizer, we consider the following eight states:

0). The recognizer is in the state	S_0	when no events, which is contained in the *epg*, has been detected.
1).	S_1	when the event a has been detected.
2).	S_2	when the event b has been detected.
3).	S_3	when the event a and b have been detected.
...		
7).	S_7	when all of the event a, b, c, and e have been detected.

We examine the conditions for each of state turning into the next state, and we get the TCCT process P_i corresponding to S_i for each $i = 0, 1, \ldots, 7$, but, here, for the time being, we ignore the handling of the time constraint of each edge.

$$\begin{aligned}
P_0 &= a.(P_0 \parallel P_1) + b.(P_0 \parallel P_2) \\
P_1 &= b.(P_1 \parallel P_3) \\
P_2 &= a.(P_2 \parallel P_3) + e.(P_2 \parallel P_4) + d \\
P_3 &= c.(P_3 \parallel P_5) + e.(P_3 \parallel P_6) + d \\
&\cdots \\
P_7 &= \alpha_{epg}
\end{aligned}$$

In the above P_0 waits for the event a or b, and has the form of $a.P' + b.Q'$. Namely when, for example, an event a occurs P_0 splits into P_0 and P_1. Here the reason why P_0 does not turn into only P_1 is to search for all possible matchings.

The output event α_{epg} indicates that a matching is completed.

Now, recall that with the *epg* the time constraint 5 on the edge (n_1, n_3) specifies that, once an event a for the node n_1 is detected, so must be c for n_3 within 5 time units. TCCT can handle properly these time constraints by setting timers, say T_1, to 5 on detecting a and by releasing it on detecting c. In the state S_3, the recognizer must handle a time constraint on each of the edges (n_1, n_3) and (n_2, n_4). For instance, if c occurs, the recognizer should release the constraint on (n_1, n_3), but not the other. If e occurs, it should do the other way around. The timer mechanism of TCCT works in this way.

In order to handle time constraints, P_i's are modified:

$$
\begin{array}{lcl}
P_0 & = & a.(P_0 \parallel P_1(T_1)^{\{T_1:5\}}) + b.(P_0 \parallel P_2(T_2)^{\{T_2:6\}}) \\
P_1(T) & = & b.(P_1(T) \parallel P_3(T, T_2)^{\{T_2:6\}}) \\
P_2(T') & = & a.(P_2(T') \parallel P_3(T_1, T')^{\{T_1:5\}}) + e.(P_2(T') \parallel \mathit{off}\langle T'\rangle.P_4) + d \\
P_3(T, T') & = & c.(P_3(T, T') \parallel \mathit{off}\langle T_1\rangle.P_5(T')) + e.(P_3(T, T') \parallel \mathit{off}\langle T'\rangle.P_6(T)) + d \\
P_4 & = & a.(P_4 \parallel P_6(T_1)^{\{T_1:5\}}) + d \\
P_5(T') & = & e.(P_5(T') \parallel \mathit{off}\langle T'\rangle.P_7) \\
P_6(T) & = & c.(P_6(T) \parallel \mathit{off}\langle T\rangle.P_7) + d \\
P_7 & = & \alpha_{epg}
\end{array}
$$

where we can show that the process P_7 outputs the signal α_{epg} each time when a matching is completed.

Given the input sequence $\langle 2\rangle \cdot b \cdot \langle 3\rangle \cdot a \cdot \langle 1\rangle \cdot e \cdot \langle 3\rangle \cdot c \cdot \langle 1\rangle \cdot d \cdot \ldots$, we get the output sequence $\langle 9\rangle \cdot \alpha_{epg} \cdot \langle 1\rangle \cdot \ldots$ as follows.

$$
\begin{array}{ll}
 & P_0 \\
\stackrel{2}{\rightarrow} & P_0 \\
\stackrel{b}{\rightarrow} & P_0 \parallel P_2(T_2)^{\{T_2:6\}} \\
\stackrel{3}{\rightarrow} & P_0 \parallel P_2(T_2)^{\{T_2:3\}} \\
\stackrel{a}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:5\}} \parallel (P_2(T_2) \parallel P_3(T_1, T_2)^{\{T_1:5\}})^{\{T_2:3\}} \\
\equiv & P_0 \parallel P_1(T_1)^{\{T_1:5\}} \parallel P_2(T_2)^{\{T_2:3\}} \parallel (P_3(T_1, T_2)^{\{T_1:5\}})^{\{T_2:3\}} \\
\stackrel{1}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:4\}} \parallel P_2(T_2)^{\{T_2:2\}} \parallel (P_3(T_1, T_2)^{\{T_1:4\}})^{\{T_2:2\}} \\
\stackrel{e}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:4\}} \parallel (P_2(T_2) \parallel \mathit{off}\langle T_2\rangle.P_4)^{\{T_2:2\}} \parallel \\
 & ((P_3(T_1, T_2) \parallel \mathit{off}\langle T_2\rangle.P_6(T_1))^{\{T_1:4\}})^{\{T_2:2\}} \\
\equiv & P_0 \parallel P_1(T_1)^{\{T_1:4\}} \parallel P_2(T_2)^{\{T_2:2\}} \parallel (\mathit{off}\langle T_2\rangle.P_4)^{\{T_2:2\}} \parallel (P_3(T_1, T_2)^{\{T_1:4\}})^{\{T_2:2\}} \parallel \\
 & ((\mathit{off}\langle T_2\rangle.P_6(T_1))^{\{T_2:2\}})^{\{T_1:4\}} \\
\stackrel{\tau*}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:4\}} \parallel P_2(T_2)^{\{T_2:2\}} \parallel P_4 \parallel (P_3(T_1, T_2)^{\{T_1:4\}})^{\{T_2:2\}} \parallel P_6(T_1)^{\{T_1:4\}} \\
\stackrel{2}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:2\}} \parallel P_2(T_2)^{\{T_2:0\}} \parallel P_4 \parallel (P_3(T_1, T_2)^{\{T_1:2\}})^{\{T_2:0\}} \parallel P_6(T_1)^{\{T_1:2\}} \\
\stackrel{\tau*}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:2\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:2\}} \\
\stackrel{1}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:1\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:1\}} \\
\stackrel{c}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:1\}} \parallel P_4 \parallel (P_6(T_1) \parallel \mathit{off}\langle T_1\rangle.P_7)^{\{T_1:1\}} \\
\stackrel{\equiv}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:1\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:1\}} \parallel (\mathit{off}\langle T_1\rangle.P_7)^{\{T_1:1\}} \\
\stackrel{\tau}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:1\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:1\}} \parallel \alpha_{epg} \\
\stackrel{\alpha_{epg}}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:1\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:1\}} \\
\stackrel{1}{\rightarrow} & P_0 \parallel P_1(T_1)^{\{T_1:0\}} \parallel P_4 \parallel P_6(T_1)^{\{T_1:0\}} \\
\stackrel{\tau*}{\rightarrow} & P_0 \parallel P_4 \\
\stackrel{d}{\rightarrow} & P_0 \\
 & \ldots
\end{array}
$$

where $\stackrel{\tau*}{\rightarrow}$ is the transitive reflexive closure of $\stackrel{\tau}{\rightarrow}$.

3 Behavioural Equivalence

Based on the conventional notion of (strong) bisimulation, we build an equivalence relation '$\sim$' over TCCT processes.

We show that '$\sim$' is a 'congruence' relation, in the sense that two bisimilar processes without any free timer-names are substitutive under all the combinators. Therefore if P and Q are admissible and bisimilar, then they are interchangeable in any contexts.

Definition 5 *(Bisimulation) A binary relation $\mathcal{R}$ over processes is a bisimulation, if $(P,Q) \in \mathcal{R}$ implies for all $\ell \in EV_{IN} \cup EV_{OUT} \cup Time \cup \{\tau\}$*

$$\textit{Whenever } P \xrightarrow{\ell} P', \textit{ for some } Q',\ Q \xrightarrow{\ell} Q' \textit{ and } (P',Q') \in \mathcal{R}, \textit{ and vice versa}$$

Definition 6 *$P, Q \in \mathcal{P}$ are bisimilar, written $P \sim Q$, if $(P,Q) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$.*

Theorem 1 ***(Congruence 1)*** *Let P, Q be processes which contain no occurrence of free timer-name and $P \sim Q$. Then*

1. $\ell.P \sim \ell.Q$ *where* $\ell \in EV_{IN} \cup EV_{OUT}$.
2. $P + R \sim Q + R$.
3. $P \parallel R \sim Q \parallel R$.
4. $P^{\{T:t\}}R \sim Q^{\{T:t\}}R$.
5. $R^{\{T:t\}}P \sim R^{\{T:t\}}Q$.

Proof
All the proofs except the one of 4 and 5 are almost the same as usual.
By the assumption $P \sim Q$, there exists a bisimulation $\mathcal{R}_0$ such that $(P,Q) \in \mathcal{R}_0$.

1. Let $\mathcal{R} \stackrel{def}{=} \mathcal{R}_0 \cup \{(\ell.P, \ell.Q)\}$.
   It is clear that $\mathcal{R}$ is a bisimulation and $(\ell.P, \ell.Q) \in \mathcal{R}$.
2. Let $\mathcal{R} \stackrel{def}{=} \mathcal{R}_0 \cup \{(O,O) \mid O \in \mathcal{P}\} \cup \bigcup_t \{((P+R)^{-t}, (Q+R)^{-t})\}$, where $t \in Time$ and P^{-t} is the process into which P would evolve with only time progress $\xrightarrow{t}$(i.e. $P \xrightarrow{t} P^{-t}$.)
   Clearly $\mathcal{R}$ is a bisimulation and $(P+R, Q+R) \in \mathcal{R}$.
3. Let $\mathcal{R} \stackrel{def}{=} \{(P' \parallel O, Q' \parallel O) \mid (P',Q') \in \mathcal{R}_0, O \in \mathcal{P}\}$.
   Clearly $\mathcal{R}$ is a bisimulation and $(P \parallel R, Q \parallel R) \in \mathcal{R}$.
4. Let $\mathcal{R} \stackrel{def}{=} \{(P'^{\{T':t'\}}R, Q'^{\{T':t'\}}R) \mid T' \notin \mathbf{ft}(P') \cup \mathbf{ft}(Q'), (P',Q') \in \mathcal{R}_0\} \cup Id$.
   Here we show that $\mathcal{R}$ is a bisimulation.
   If $(P'^{\{T':t'\}}, Q'^{\{T':t'\}}) \in \mathcal{R}$ and $P'^{\{T':t'\}} \xrightarrow{\ell} R'$, then there are four cases for this transition as follows, according to which rule it stems from.
   (a) (**Time Progress**)

$$\frac{P' \stackrel{s'-t'}{\rightarrow} P''}{P'^{\{T':t'\}} R \stackrel{s'-t'}{\rightarrow} P''^{\{T':s'\}} R (= R')}$$

Then $Q' \stackrel{s'-t'}{\rightarrow} Q''$ and $(P'', Q'') \in \mathcal{R}_0$ because of $(P', Q') \in \mathcal{R}_0$, hence we have $Q'^{\{T':t'\}} R \stackrel{s'-t'}{\rightarrow} Q''^{\{T':s'\}} R$ with the same rule above, and $(P''^{\{T':s'\}} R, Q''^{\{T':s'\}} R) \in \mathcal{R}$ as required.

(b) (**Event-in, Event-out**)

$$\frac{P' \stackrel{l}{\rightarrow} P''}{P'^{\{T':t'\}} R \stackrel{l}{\rightarrow} P''^{\{T':t'\}} R (= R')}$$

Then $Q' \stackrel{l}{\rightarrow} Q''$ and $(P'', Q'') \in \mathcal{R}_0$ because of $(P', Q') \in \mathcal{R}_0$, hence we have $Q'^{\{T':t'\}} R \stackrel{l}{\rightarrow} Q''^{\{T':t'\}} R$ with the same rule above, and $(P''^{\{T':t'\}} R, Q''^{\{T':t'\}} R) \in \mathcal{R}$ as required.

(c) (**Timer 1**)

$$\frac{}{P'^{\{T':0(=t')\}R} \stackrel{\tau}{\rightarrow} R}$$

This case is trivial.

(d) (**Timer 2**)

$$\frac{P' \stackrel{\tau}{\rightarrow} P''}{P'^{\{T':t'\}} R \stackrel{\tau}{\rightarrow} P''^{\{T':t'\}} R}$$

This is almost the same as **Event-in** and **Event-out**.

Note that since $((\mathit{off}\langle T'\rangle P_1')^{\{T':t'\}}, Q') \notin \mathcal{R}$, the following case is excluded above.

(**Timer 3**)

$$\frac{}{(\mathit{off}\langle T'\rangle P_1')^{\{T':0(=t')\}} R \stackrel{\tau}{\rightarrow} 0}$$

5. Let $\mathcal{R} \stackrel{def}{=} \mathcal{R}_0 \cup Id \cup \{(R'^{\{T':t'\}} P, R'^{\{T':t'\}} Q)\}$.
   Clearly $\mathcal{R}$ is a bisimulation and $(R^{\{T:t\}} P, R^{\{T:t\}} Q) \in \mathcal{R}$.

The following definitions and lemmas are employed in the proof of Theorem 2.

Definition 7 *$\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a bisimulation up to $\sim$, if $(P, Q) \in \mathcal{R}$ implies for all $\ell \in EV_{IN} \cup EV_{OUT} \cup Time \cup \{\tau\}$,*

Whenever $P \stackrel{\ell}{\rightarrow} P'$, for some Q', $Q \stackrel{\ell}{\rightarrow} Q'$ and $(P', Q') \in \sim \mathcal{R} \sim$, and vice versa

Lemma 1 *If $\mathcal{R}$ is a bisimulation up to $\sim$, then $\sim \mathcal{R} \sim$ is a bisimulation.*

Definition 8 $P \sim^{\mathcal{T}} Q$ *iff* $\exists T_i, i = 1, \ldots, n, \forall t_i, \forall R_i (i = 1, \ldots, n)$,

$$(\cdots((P^{\{T_1:t_1\}} R_1)^{\{T_2:t_2\}} R_2)\cdots)^{\{T_n:t_n\}} R_n$$

and

$$(\cdots((Q^{\{T_1:t_1\}} R_1)^{\{T_2:t_2\}} R_2)\cdots)^{\{T_n:t_n\}} R_n$$

are admissible and bisimilar.

Lemma 2 *If $P \sim^{\mathcal{T}} Q$, then for any $\{T:t\}$ and R, $P^{\{T:t\}}R \sim^{\mathcal{T}} Q^{\{T:t\}}R$.*

Theorem 2 ***(Congruence 2)*** *Let P_A, Q_B be admissible and $A \stackrel{def}{=} P_A, B \stackrel{def}{=} Q_B$. Suppose that $P_A[R/A] \sim Q_B[R/B]$ for any admissible process R, where $P_A[R/A]$ is the process obtained by replacing each occurrence of A in P_A with R. Then $A \sim B$.*

Proof Let $\mathcal{R}_0 = \{(G[A/C], G[B/C]) \mid G$ contains at most one process constant C, *and* $(\cdots((G^{\{T_1:t_1\}}R_1)^{\{T_2:t_2\}}R_2)\cdots)^{\{T_n:t_n\}}R_n$is admissible for some $(\{T_1:t_1\}, R_1), (\{T_2:t_2\}, R_2), \ldots,$ and$(\{T_n:t_n\}, R_n)\}$.

We show the following lemma:

Lemma 3 *If $G[A/C] \stackrel{\ell}{\to} P'$, for some Q' and Q'', $G[B/C] \stackrel{\ell}{\to} Q'' \sim^{\mathcal{T}} Q'$ and $(P', Q') \in \mathcal{R}_0$.*

Let $\mathcal{R} \stackrel{def}{=} \mathcal{R}_0 \cap \{(G[A/C], G[B/C]) \mid G \text{ is admissible}\}$. Then $\mathcal{R}$ is bisimulation up to $\sim$. For then, when $G = C$, $(A, B) \in \mathcal{R}$, and hence $A \sim B$.

Now let us give the proof of Lemma 3.

We prove it by induction on the depth of the inference by which $G[A/C] \stackrel{\ell}{\to} P'$ is inferred. We discuss it by cases on the form of G as follows.

- $G = C$.

  $G[A/C] = A$ and $A \stackrel{\ell}{\to} P'$. By the transition rules, $A \stackrel{\ell}{\to} P'$ must be inferred from $P_A \stackrel{\ell}{\to} P'$. By induction, $P_A[B/A] \stackrel{\ell}{\to} Q'' \sim^{\mathcal{T}} Q'$ and $(P', Q') \in \mathcal{R}_0$. By $P_A[R/A] \sim Q_B[R/B]$ for any process R, i.e. $P_A[R/A] \sim^{\mathcal{T}} Q_B[R/B]$, $Q_B[B/B] \stackrel{\ell}{\to} Q''' \sim (=\sim^{\mathcal{T}}) Q'' \sim^{\mathcal{T}} Q'$. Since $B = Q_B (= Q_B[B/B])$, $B \stackrel{\ell}{\to} Q''' \sim^{\mathcal{T}} Q'$ and $(P', Q') \in \mathcal{R}$.

- $G = \ell.G'$ where $\ell \in Ev_{IN} \cup EV_{OUT}$.

  Then $G[A/C] = \ell.G'[A/C]$. Hence $P' = G'[A/C]$ (**Event-in, Event-out**), or $\ell.G'[A/C]$ (**Time progress**). We have also $G[B/C] = \ell.G'[B/C] \stackrel{\ell}{\to} G'[B/C]$, $\ell.G'[B/C] \stackrel{t}{\to} \ell.G'[B/C]$, and $(G'[A/C], G'[B/C]), (\ell.G'[A/C], \ell.G'[B/C]) \in \mathcal{R}_0$.

- $G = G_1 + G_2$.

  Then $G[A/C] = G_1[A/C] + G_2[A/C]$, so $G[A/C] \stackrel{\ell}{\to} P'$ must be inferred by either the following two inferences.

  1. $$\frac{G_i[A/C] \stackrel{\ell}{\to} P'}{G_1[A/C] + G_2[A/C] \stackrel{\ell}{\to} P'}$$

     By induction, $G_i[B/C] \stackrel{\ell}{\to} Q'' \sim^{\mathcal{T}} Q', (P', Q') \in \mathcal{R}_0$. Hence we have the inference

     $$\frac{G_i[B/C] \stackrel{\ell}{\to} Q''}{G_1[B/C] + G_2[B/C] \stackrel{\ell}{\to} Q''}$$

  2. $$\frac{G_i[A/C] \stackrel{t}{\to} P'_i}{G_1[A/C] + G_2[A/C] \stackrel{t}{\to} P'_1 + P'_2}$$

     By induction, $G_i[B/C] \stackrel{t}{\to} Q''_i \sim^{\mathcal{T}} Q'_i, (P'_i, Q'_i) \in \mathcal{R}_0$. Hence we have the inference

     $$\frac{G_i[B/C] \stackrel{t}{\to} Q''_1}{G_1[B/C] + G_2[B/C] \stackrel{t}{\to} Q''_1 + Q''_2}$$

Clearly $(P'_1 + P'_2, Q''_1 + Q''_2) \in \mathcal{R}_0$.

- $G = G_1 \parallel G_2$.

  Almost the same as above, and we omit the proof.

- $G = G'^{\{T:t\}}$.

  There are five cases for the transition $G'^{\{T:t\}}[A/C] = G'[A/C]^{\{T:t\}} \xrightarrow{\ell} P'$ as follows, according to which rule it stems from.

  1. (**Time Progress**)

     $$\frac{G'[A/C] \xrightarrow{s-t} P''}{G'[A/C]^{\{T:t\}}R \xrightarrow{s-t} P''^{\{T:s\}}R(= P')}$$

     By induction, $G'[B/C] \xrightarrow{s-t} Q''' \sim^{\mathcal{T}} Q'', (P'', Q'') \in \mathcal{R}_0$.
     By Lemma 2, $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$.
     Therefore $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$ and $(P''^{\{T:t\}}R, Q''^{\{T:t\}}R) \in \mathcal{R}_0$, and

     $$\frac{G'[B/C] \xrightarrow{s-t} Q'''}{G'[B/C]^{\{T:t\}}R \xrightarrow{s-t} Q'''^{\{T:s\}}R}$$

     as required.

  2. (**Event-in, Event-out**)

     $$\frac{G'[A/C] \xrightarrow{l} P''}{G'[A/C]^{\{T:t\}}R \xrightarrow{l} P''^{\{T:t\}}R(= P')}$$

     By induction, $G'[B/C] \xrightarrow{l} Q''' \sim^{\mathcal{T}} Q'', (P'', Q'') \in \mathcal{R}_0$.
     By Lemma 2, $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$.
     Therefore $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$ and $(P''^{\{T:t\}}R, Q''^{\{T:t\}}R) \in \mathcal{R}_0$, and

     $$\frac{G'[B/C] \xrightarrow{l} Q'''}{G'[B/C]^{\{T:t\}}R \xrightarrow{l} Q'''^{\{T:t\}}R}$$

     as required.

  3. (**Timer 1**)

     $$\frac{}{G'[A/C]^{\{T:0\}}R \xrightarrow{\tau} R(= P')}$$

     It is trivial.

  4. (**Timer 2**)

     $$\frac{}{(\mathit{off}\langle T\rangle.G''[A/C])^{\{T:t\}}R \xrightarrow{\tau} G''[A/C](= P')}$$

     Then $(\mathit{off}\langle T\rangle.G''[B/C])^{\{T:t\}}R \xrightarrow{\tau} G''[B/C]$, and $(G''[A/C], G''[B/C]) \in \mathcal{R}_0$ as required.

  5. (**Timer 3**)

$$\frac{G'[A/C] \xrightarrow{\tau} P''}{G'[A/C]^{\{T:t\}}R \xrightarrow{\tau} P''^{\{T:t\}}R (= P')}$$

By induction, $G'[B/C] \xrightarrow{\tau} Q''' \sim^{\mathcal{T}} Q''$, $(P'', Q'') \in \mathcal{R}_0$.
By Lemma 2, $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$.
Therefore $Q'''^{\{T:t\}}R \sim^{\mathcal{T}} Q''^{\{T:t\}}R$ and $(P''^{\{T:t\}}R, Q''^{\{T:t\}}R) \in \mathcal{R}_0$, and

$$\frac{G'[B/C] \xrightarrow{\tau} Q'''}{G'[B/C]^{\{T:t\}}R \xrightarrow{\tau} Q'''^{\{T:t\}}R}$$

as required.

4 Discussion: Separation of time and actions

With TCCT, passage of time is modeled by time progress transition, and execution of actions is supposed to be instantaneous. As discussed in [9], this separation of time and actions makes the theoretical treatment simpler and less stressful, while it introduces eccentric processes which diverge without time progress or are stuck without applicable rules.

Suppose that a process constant B has the defining equation $B \stackrel{def}{=} (\mathit{off}\langle T\rangle.B)^{\{T:t\}})$. It infinitely repeats internal actions of set and release of the timer T, which does not consume time at all:

$$B(\equiv (\mathit{off}\langle T\rangle.B)^{\{T:t\}}) \xrightarrow{\tau} B \xrightarrow{\tau} B \xrightarrow{\tau} \cdots$$

This kind of divergency generally arises in timed calculi that separate time from actions. In Timed CCS, for instance, the following agent (or process in TCCT) infinitely and instantaneously repeats inter-process communication: $(recX.\alpha@t.X) \parallel (recX.\overline{\alpha}@t.X) \parallel \epsilon(d).P$.

One possible solution to avoid such unrealistic transition is to model actions with time progress. A timed process algebra ACSR [6] has synchronous timed actions, each of which represents the usage of some 'resource' for a single time unit. As the same way, we could obtain another TCCT where it takes time to execute actions.

Another solution, which we adopt for TCCT, is to keep the separation of time and actions and cope with the problem by treating only 'weakly-guarded' constants. The constant B above, for example, is not weakly-guarded, because the constant B is not prefixed by any input event (but by $\mathit{off}\langle T\rangle$) in its defining equation. Note that, if we weaken Def. 5 of weakly-guardedness by changing 'prefixed by some input event' to 'prefixed by some input event *or* $\mathit{off}\langle\rangle$', B becomes no longer weakly-guarded. This is why we adopt the current definition for weakly-guardedness.

The problem above is not particular to TCCT but is common with other timed process calculi. The phenomena are essentially derived from that time passage is represented by a transition. In TCCT we exclude such pathological processes with 'admissibility' and 'weakly-guardedness'. There may be better solutions.

References

[1] J. C. M. Baeten and J. A. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3:142–188, 1991.

[2] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabitilies. In IEEE Computer Society Press, editor, *Proceedings of the Real-Time*

Systems Symposium - 1990, pages 278–287, Lake Buena Vista, Florida, USA, December 1990. IEEE Computer Society Press.

[3] Y. Hayashi, A. Itou, R. Nakajima, and M. Tanabe. Timed event pattern matching. Submitted for publication.

[4] Y. Hayashi, T. Izumida, N. Kawakatsu, R. Nakajima, and M. Tanabe. Timed pattern matching - formalism, machines and applications -(in Japanese). *Computer Software*, 17(5), 2000.

[5] Matthew Hennessy and Tim Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, March 1995.

[6] I. Lee, H. Ben-Abdallah, and J. Choi. *A Process Algebraic Method for Real-time Systems*, chapter 7. John Wiley & Sons, 1996.

[7] R. Milner. *A Calculus of Communicating Systems*. Springer, Berlin, 1 edition, 1980.

[8] Faron Moller and Chris Tofts. A temporal calculus of communicating systems. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90: Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 401–415, Amsterdam, The Netherlands, 27–30August 1990. Springer-Verlag.

[9] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 526–548, Berlin, Germany, June 1992. Springer.

[10] X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114(1):131–178, 1994.

[11] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 549–572, Berlin, Germany, June 1992. Springer.

[12] George M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. In Laurent Kott, editor, *Automata, Languages and Programming, 13th International Colloquium*, volume 226 of *Lecture Notes in Computer Science*, pages 314–323, Rennes, France, 15–19 July 1986. Springer-Verlag.

[13] Wang Yi. CCS + time = an interleaving model for real time systems. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium*, volume 510 of *Lecture Notes in Computer Science*, pages 217–228, Madrid, Spain, 8–12 July 1991. Springer-Verlag.