Timed process calculi: from durationless actions to durational ones ⋆

Marco Bernardo[1] Flavio Corradini[2] Luca Tesei[2]

[1] Dipartimento di Scienze di Base e Fondamenti, Università di Urbino, Italy
[2] Scuola di Scienze e Tecnologie, Università di Camerino, Italy

Abstract. Several timed process calculi have been proposed in the literature, which mainly differ for the way in which delays are represented. In particular, a distinction is made between integrated-time calculi, in which actions are durational, and orthogonal-time calculi, in which actions are instantaneous and delays are expressed separately. To reconcile the two approaches, in a previous work an encoding from the integrated-time calculus CIPA to the orthogonal-time calculus TCCS was defined, which preserves timed bisimilarity. To complete the picture, in this paper we consider the reverse translation, by examining the modifications to the two calculi that are needed to make an encoding feasible, as well as the behavioral equivalence that is appropriate to preserve. We then introduce an encoding from modified TCCS to modified CIPA, and show that it can only preserve the weak variant of timed bisimilarity.

1 Introduction

Computing systems are characterized not only by their functional behavior, but also by their quantitative features. In particular, *timing aspects* play a fundamental role, as they describe the temporal evolution of system activities. This is especially true for *real-time systems*, which are considered correct only if the execution of their activities fulfills certain *temporal constraints*.

When modeling these systems, time is represented through nonnegative numbers. In the following, we refer to *abstract time*, in the sense that we use time as a parameter for expressing constraints about instants of occurrences of actions. Unlike *physical time*, abstract time permits simplifications that are convenient, on the conceptual side, to obtain tractable models.

Many *timed process calculi* have appeared in the literature. Among them, we mention temporal CCS [8], timed CCS [15], timed CSP [13], real-time ACP [2], urgent LOTOS [4], CIPA [1], TPL [7], ATP [11], TIC [12], and PAFAS [6]. As observed in [10, 14, 5], these calculi differ on the basis of a number of time-related options, some of which are recalled below:

– *Durationless actions* versus *durational actions*. In the first case, actions are instantaneous events and time passes in between them; hence, functional

⋆ Work partially supported by the MIUR-PRIN project CINA.

behavior and time are *orthogonal*. In the second case, every action takes a fixed amount of time to be performed and time passes only due to action execution; hence, functional behavior and time are *integrated*.
- *Relative time* versus *absolute time*. Assume that timestamps are associated with the events observed during system execution. In the first case, each timestamp refers to the time instant of the previous observation. In the second case, all timestamps refer to the starting time of the system execution.
- *Global clock* versus *local clocks*. In the first case, there is a single clock that governs time passing. In the second case, there are several clocks associated with the various system parts, which elapse independent of each other although they define a unique notion of global time.

Moreover, for timed process calculi, there are several different interpretations of action execution, in terms of whether and when it can be delayed, such as:

- *Eagerness*: actions must be performed as soon as they become enabled, i.e., without any delay, thereby implying that they are urgent.
- *Laziness*: after getting enabled, actions can be delayed arbitrarily long before they are executed.
- *Maximal progress*: enabled actions can be delayed arbitrarily long unless they are involved in synchronizations, in which case they are urgent.

In this paper, we focus on two different timed process calculi obtained by suitably combining the time-related options mentioned above. More precisely, the first calculus, TCCS [8], is inspired by the *two-phase functioning principle*, according to which actions are durationless, time is relative, and there is a single global clock. In contrast, the second calculus, CIPA [1], is inspired by the *one-phase functioning principle*, according to which actions are durational, time is absolute, and several local clocks are present.

In [5], it was shown that some of the choices concerned with the time-related options and action execution interpretations are not irreconcilable, thus permitting the interchange of concepts and analysis techniques. More precisely, the different expressive power of the two considered process calculi was investigated by developing a bisimulation-semantics-preserving encoding of CIPA processes into TCCS processes for each action execution interpretation.

In this paper, we complete the previous expressiveness study by considering the reverse encoding from TCCS processes to CIPA processes, which may also be exploited for checking bisimilarity of TCCS processes more efficiently. As pointed out at the end of [5], there are several issues that need to be addressed before the reverse encoding can be established. Our first contribution is to provide a solution for each of the various problems. Our second contribution is the definition of the reverse encoding, together with a full abstraction result of this reverse encoding under *weak* timed bisimilarity, as opposed to the direct encoding demonstrated to be fully abstract with respect to *strong* timed bisimilarity in [5].

The rest of the paper is organized as follows. In Sect. 2, we recall TCCS and CIPA. In Sect. 3, we discuss the main design decisions behind the reverse encoding. In Sect. 4, we define the reverse encoding and show that it preserves weak timed bisimilarity. Finally, in Sect. 5 we provide some concluding remarks.

2 Background

2.1 Preliminaries

We denote by A a nonempty set of visible actions – ranged over by a, b – and by $\bar{A} = \{\bar{a} \mid a \in A\}$ the set of corresponding coactions such that $\bar{\bar{a}} = a$ for all $a \in A$. We use $Act = A \cup \bar{A} \cup \{\tau\}$ to indicate the set of all actions – ranged over by α, β – where τ is the invisible action.

We denote by *Rel* a set of action relabeling functions. Each such function $\varphi : Act \to Act$ satisfies $\varphi(\tau) = \tau$ and $\overline{\varphi(a)} = \varphi(\bar{a})$ for all $a \in Act \setminus \{\tau\}$.

We denote by $\mathcal{T} = (T, \boxplus, \sqsubseteq)$ a time domain such that $T \cap Act = \emptyset$, which is equipped with an associative operation $\boxplus$ possessing neutral element and a total order relation $\sqsubseteq$ satisfying $t_1 \sqsubseteq t_2$ iff there exists $t' \in T$ such that $t_1 \boxplus t' = t_2$. Typical choices are $T = \mathbb{N}$ and $T = \mathbb{R}_{\geq 0}$, with the usual $+$ and $\leq$.

Finally, we denote by *Var* a nonempty set of process variables – ranged over by X, Y – whose occurrences can be free or bound by "rec".

2.2 Durationless Actions: TCCS

We recall from [8] the syntax of TCCS. As in [5], we leave out the idling operator δ and the weak choice operator $\oplus$, as they have no direct counterpart in CIPA.

Definition 1. *The set of process terms of the process language $\mathcal{PL}_{\rm TCCS}$ is generated by the following syntax:*

$P ::= \mathbf{0}$		*stopped process*
$\mid \alpha.P$		*action prefix*
$\mid (t).P$		*delay prefix*
$\mid P + P$		*alternative composition*
$\mid P \mid P$		*parallel composition*
$\mid P \backslash L$		*restriction*
$\mid P[\varphi]$		*relabeling*
$\mid X$		*process variable*
$\mid \mathrm{rec}\, X : P$		*recursion*

where $\alpha \in Act$, $t \in \mathbb{N}_{>0}$, $L \subseteq A$, $\varphi \in Rel$, and $X \in Var$. We denote by $\mathbb{P}_{\rm TCCS}$ the set of closed and guarded process terms of $\mathcal{PL}_{\rm TCCS}$. ■

Process $\mathbf{0}$ can neither proceed with any action, nor proceed through time. Process $\alpha.P$ can perform instantaneous action α and then evolves into process P; action α is urgent, hence time cannot progress before α is executed. Process $(t).P$ evolves into process P after a delay equal to t.

Process $P_1 + P_2$ represents a nondeterministic choice between processes P_1 and P_2, with the choice being resolved depending on whether an action of P_1 or P_2 is executed first. Time does not resolve choices, in the sense that any initial passage of time common to P_1 and P_2 must be allowed without making the choice. Process $P_1 | P_2$ describes the parallel composition of processes P_1 and P_2,

$\dfrac{}{\alpha.P \stackrel{\alpha}{\longrightarrow} P}$	$\dfrac{}{(t).P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P}$
$\dfrac{P_1 \stackrel{\alpha}{\longrightarrow} P_1'}{P_1 + P_2 \stackrel{\alpha}{\longrightarrow} P_1'} \quad \dfrac{P_2 \stackrel{\alpha}{\longrightarrow} P_2'}{P_1 + P_2 \stackrel{\alpha}{\longrightarrow} P_2'}$	$\dfrac{}{(t + t').P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} (t').P} \quad \dfrac{P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'}{(t').P \stackrel{t+t'}{\longrightarrow\!\!\!\!\!\sim} P'}$
$\dfrac{P_1 \stackrel{\alpha}{\longrightarrow} P_1'}{P_1 \vert P_2 \stackrel{\alpha}{\longrightarrow} P_1' \vert P_2} \quad \dfrac{P_2 \stackrel{\alpha}{\longrightarrow} P_2'}{P_1 \vert P_2 \stackrel{\alpha}{\longrightarrow} P_1 \vert P_2'}$	$\dfrac{P_1 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_1' \quad P_2 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_2'}{P_1 + P_2 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_1' + P_2'}$
$\dfrac{P_1 \stackrel{a}{\longrightarrow} P_1' \quad P_2 \stackrel{\bar{a}}{\longrightarrow} P_2'}{P_1 \vert P_2 \stackrel{\tau}{\longrightarrow} P_1' \vert P_2'}$	$\dfrac{P_1 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_1' \quad P_2 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_2'}{P_1 \vert P_2 \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P_1' \vert P_2'}$
$\dfrac{P \stackrel{\alpha}{\longrightarrow} P' \quad \alpha \notin L \cup \bar{L}}{P\backslash L \stackrel{\alpha}{\longrightarrow} P'\backslash L}$	$\dfrac{P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'}{P\backslash L \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'\backslash L}$
$\dfrac{P \stackrel{\alpha}{\longrightarrow} P'}{P[\varphi] \stackrel{\varphi(\alpha)}{\longrightarrow} P'[\varphi]}$	$\dfrac{P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'}{P[\varphi] \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'[\varphi]}$
$\dfrac{P\{\mathrm{rec}\, X : P \hookrightarrow X\} \stackrel{\alpha}{\longrightarrow} P'}{\mathrm{rec}\, X : P \stackrel{\alpha}{\longrightarrow} P'}$	$\dfrac{P\{\mathrm{rec}\, X : P \hookrightarrow X\} \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'}{\mathrm{rec}\, X : P \stackrel{t}{\longrightarrow\!\!\!\!\!\sim} P'}$

Table 1. Structural operational semantic rules for TCCS

where any two complementary actions may synchronize thereby resulting in a τ action; also in this case, any initial passage of time must be permitted.

Process $P\backslash L$ behaves as process P except for actions in $L \cup \bar{L}$, which are forbidden; this operator is useful to force synchronizations between complementary actions. Process $P[\varphi]$ behaves as process P, with the difference that every performed action is transformed via φ; this operator allows processes with different actions to communicate. Finally, $\mathrm{rec}\, X : P$ represents a recursive process, which behaves as process P in which every free occurrence of X is replaced by $\mathrm{rec}\, X : P$ itself; the resulting process will be denoted by $P\{\mathrm{rec}\, X : P \hookrightarrow X\}$.

Following [8], the intuitive meaning of process terms is formalized in Table 1. Transition relation $\longrightarrow$ on the left represents the functional behavior. Transition relation $\stackrel{t}{\longrightarrow\!\!\!\!\!\sim}$ on the right represents the timing behavior according to time additivity (second and third rules) and time determinism (fourth and fifth rules); the second rule is necessary for the applicability of the fourth and fifth ones, while the third rule is necessary for the forthcoming equivalence.

A notion of weak bisimilarity for TCCS was studied in [9]. It is an extension of Milner's weak bisimilarity that is capable of summing up delays while abstracting from τ actions. Weak transitions are defined as follows:

– $\Longrightarrow = (\stackrel{\tau}{\longrightarrow})^*$.
– $\stackrel{a}{\Longrightarrow} = \Longrightarrow \stackrel{a}{\longrightarrow} \Longrightarrow$.
– $\stackrel{\hat{\alpha}}{\Longrightarrow} = \Longrightarrow$ if $\alpha = \tau$, $\stackrel{\hat{\alpha}}{\Longrightarrow} = \stackrel{\alpha}{\Longrightarrow}$ if $\alpha \neq \tau$.
– $\stackrel{t}{\Longrightarrow} = \Longrightarrow \stackrel{t_1}{\rightsquigarrow} \Longrightarrow \cdots \Longrightarrow \stackrel{t_n}{\rightsquigarrow} \Longrightarrow$ where $t = \sum_{1 \leq i \leq n} t_i$, $n \in \mathbb{N}_{\geq 1}$.

Definition 2. *A symmetric relation $\mathcal{B}$ over $\mathbb{P}_{\rm TCCS}$ is a* weak timed bisimulation *iff, whenever $(P_1, P_2) \in \mathcal{B}$, then for all actions $\alpha \in Act$ and delays $t \in \mathbb{N}_{>0}$:*

– *For each $P_1 \stackrel{\alpha}{\longrightarrow} P_1'$ there exists $P_2 \stackrel{\hat{\alpha}}{\Longrightarrow} P_2'$ such that $(P_1', P_2') \in \mathcal{B}$.*
– *For each $P_1 \stackrel{t}{\rightsquigarrow} P_1'$ there exists $P_2 \stackrel{t}{\Longrightarrow} P_2'$ such that $(P_1', P_2') \in \mathcal{B}$.*

$P_1 \approx_{\rm TCCS} P_2$ iff (P_1, P_2) is contained in a weak timed bisimulation. ■

2.3 Durational Actions: CIPA

We recall from [1] the syntax of CIPA. As in [5], we add the relabeling operator.

Definition 3. *The set of process terms of the process language $\mathcal{PL}_{\rm CIPA}$ is generated by the following syntax:*

Q	::=	nil	*inactive process*
	|	$a.Q$	*durational action prefix*
	|	wait $t.Q$	*waiting prefix*
	|	$Q + Q$	*alternative composition*
	|	$Q \vert Q$	*parallel composition*
	|	$Q \backslash L$	*restriction*
	|	$Q[\varphi]$	*relabeling*
	|	X	*process variable*
	|	rec $X : Q$	*recursion*

where $a \in Act \setminus \{\tau\}$, $t \in \mathbb{N}_{>0}$, $L \subseteq A$, $\varphi \in Rel$, and $X \in Var$. We denote by $\mathbb{P}_{\rm CIPA}$ the set of closed and guarded process terms of $\mathcal{PL}_{\rm CIPA}$. ■

Process nil cannot proceed with any action, but can let time pass. Process $a.Q$ can perform urgent action a and evolves into process Q after the execution of a has finished; all occurrences of an action are assumed to have the same duration, which is established by a function $\Delta : (Act \setminus \{\tau\}) \to \mathbb{N}_{>0}$ such that $\Delta(\bar{a}) = \Delta(a)$. Process wait $t.Q$ waits for time t and then becomes process Q. All the other operators work as expected, with the additional constraints that each relabeling function φ must preserve durations, i.e., $\Delta(\varphi(a)) = \Delta(a)$ for all $a \in Act \setminus \{\tau\}$, and any pair of actions a and $\bar{a}$ can synchronize only if they start at the same time, yielding a τ action with the same duration as the two original actions.

Following [1], the set $\mathbb{KP}$ of states correspond to process terms augmented with local clocks, so to keep track of the time elapsed in the various sequential components. The shorthand $t \Rightarrow Q$ means that the clock value $t \in \mathbb{N}_{\geq 0}$ is distributed over all subprocesses of Q according to the extended syntax for $\mathbb{KP}$:

$$\frac{}{t \Rightarrow a.Q \xrightarrow[\Delta(a)]{a@t} (t + \Delta(a)) \Rightarrow Q} \qquad \frac{}{t \Rightarrow \mathrm{wait}\, t'.Q \xrightarrow[t']{\tau@t} (t + t') \Rightarrow Q}$$

$$\frac{K_1 \xrightarrow[d]{\alpha@t} K_1' \quad \neg(K_2 \xrightarrow[d']{\alpha'@t'} K_2' \wedge t' < t)}{K_1 + K_2 \xrightarrow[d]{\alpha@t} K_1'} \qquad \frac{K_2 \xrightarrow[d]{\alpha@t} K_2' \quad \neg(K_1 \xrightarrow[d']{\alpha'@t'} K_1' \wedge t' < t)}{K_1 + K_2 \xrightarrow[d]{\alpha@t} K_2'}$$

$$\frac{K_1 \xrightarrow[d]{\alpha@t} K_1' \quad \neg(K_2 \xrightarrow[d']{\alpha'@t'} K_2' \wedge t' < t)}{K_1|K_2 \xrightarrow[d]{\alpha@t} K_1'|K_2} \qquad \frac{K_2 \xrightarrow[d]{\alpha@t} K_2' \quad \neg(K_1 \xrightarrow[d']{\alpha'@t'} K_1' \wedge t' < t)}{K_1|K_2 \xrightarrow[d]{\alpha@t} K_1|K_2'}$$

$$\frac{K_1 \xrightarrow[d]{a@t} K_1' \quad K_2 \xrightarrow[d]{\bar{a}@t} K_2'}{K_1|K_2 \xrightarrow[d]{\tau@t} K_1'|K_2'} \qquad \frac{K \xrightarrow[d]{\alpha@t} K' \quad \alpha \notin L \cup \bar{L}}{K\backslash L \xrightarrow[d]{\alpha@t} K'\backslash L}$$

$$\frac{K \xrightarrow[d]{\alpha@t} K'}{K[\varphi] \xrightarrow[d]{\varphi(\alpha)@t} K'[\varphi]} \qquad \frac{t \Rightarrow Q\{\mathrm{rec}\, X : Q \hookrightarrow X\} \xrightarrow[d]{\alpha@t} K'}{t \Rightarrow \mathrm{rec}\, X : Q \xrightarrow[d]{\alpha@t} K'}$$

Table 2. Structural operational semantic rules for CIPA

$$K ::= t \Rightarrow \mathrm{nil} \mid t \Rightarrow a.Q \mid t \Rightarrow \mathrm{wait}\, t'.Q \mid t \Rightarrow \mathrm{rec}\, X : Q \mid K + K \mid K|K \mid K\backslash L \mid K[\varphi]$$

In this setting, any transition is of the form $K \xrightarrow[d]{\alpha@t} K'$, meaning that $K \in \mathbb{KP}$ performs an action of name $\alpha \in Act$ that starts at time $t \in \mathbb{N}_{\geq 0}$ and has duration $d \in \mathbb{N}_{>0}$, after which evolves to $K' \in \mathbb{KP}$. The transition relation is defined in Table 2, where negative premises are present as in [5]. Those in the rules for alternative composition enforce action urgency. Those in the rules for parallel composition avoid the generation of *ill-timed paths*, i.e., computations along which the starting time of some actions decreases as the execution proceeds.

A notion of weak bisimilarity for CIPA was studied in [1] under the name of timed branching bisimilarity, which has the capability of summing up consecutive waitings. Weak transitions are defined as follows: $\Longrightarrow = \xrightarrow[d_1]{\tau@t_1} \cdots \xrightarrow[d_n]{\tau@t_n}$, $n \in \mathbb{N}$.

Definition 4. *A symmetric relation $\mathcal{B}$ over $\mathbb{KP}$ is a* weak timed bisimulation *iff, whenever $(K_1, K_2) \in \mathcal{B}$, then for all actions $\alpha \in Act$, starting times $t \in \mathbb{N}_{\geq 0}$, and durations $d \in \mathbb{N}_{>0}$ it holds that for each $K_1 \xrightarrow[d]{\alpha@t} K_1'$:*

- *When $\alpha \neq \tau$, there exists $K_2 \Longrightarrow K_2'' \xrightarrow[d]{\alpha@t} K_2'$ such that $(K_1, K_2'') \in \mathcal{B}$ and $(K_1', K_2') \in \mathcal{B}$.*

– *When* $\alpha = \tau$, *either* $(K_1', K_2) \in \mathcal{B}$, *or there exists* $K_2 \Longrightarrow K_2'' \xrightarrow[d']{\alpha@t'} K_2'$ *such that* $(K_1, K_2'') \in \mathcal{B}$ *and* $(K_1', K_2') \in \mathcal{B}$.

$K_1 \approx_{\rm CIPA} K_2$ *iff* (K_1, K_2) *is contained in a weak timed bisimulation. Moreover,* $Q_1 \approx_{\rm CIPA} Q_2$ *iff* $(0 \Rightarrow Q_1, 0 \Rightarrow Q_2)$ *is contained in a weak timed bisimulation.* ■

Although in the clause $\alpha = \tau$ it may be $t' \neq t$ and $d' \neq d$, possible subsequent visible actions must start at the same time in both processes for $\approx_{\rm CIPA}$ to hold.

3 Design of the Reverse Encoding

In [5], where an encoding from CIPA to TCCS was proposed, a number of issues were raised about the existence of a reverse encoding from TCCS back to CIPA. In this section, we recall those issues and discuss how to address them.

3.1 Adapting TCCS and CIPA

The first issue is related to the range of values that can be used in CIPA to express action durations. In TCCS, it is possible to describe both timed processes and untimed ones. Consider for example the untimed TCCS process $a.b.\mathbf{0}$. This cannot be translated into a reasonably corresponding CIPA process for the very simple reason that actions a and b are instantaneous, but CIPA does not allow zero durations. Moreover, due to instantaneous actions, TCCS processes may exhibit Zeno behaviors, which are not possible in CIPA. For instance, the timed TCCS process $(t_1).a.\mathrm{rec}\, X : (b.X + c.(t_2).\mathbf{0})$ may perform, after time t_1 and action a, an arbitrary (even infinite) number of actions b at the same time. These problems can be straightforwardly solved by admitting zero durations in CIPA through an extended duration function $\Delta : (Act \setminus \{\tau\}) \to \mathbb{N}$.

The second issue that we address is timelock. In a TCCS process, time does not solve choices; indeed, the operational rules for alternative and parallel composition allow time to pass only if all the subprocesses do so. As a consequence, a local timelock always implies a global timelock, which may in turn determine a deadlock. By contrast, in CIPA timelock cannot occur unless there is a deadlock, because time passing is associated with action execution and explicit waiting. Consider the TCCS process $\mathbf{0}+(t).\mathbf{0}$ and the ideally corresponding CIPA process $\mathrm{nil} + \mathrm{wait}\, t.\mathrm{nil}$; the former process cannot let time pass, while the latter process can. The same would happen with $(a.\mathbf{0})\backslash\{a\} + (t).\mathbf{0}$ and $(a.\mathrm{nil})\backslash\{a\} + \mathrm{wait}\, t.\mathrm{nil}$.

To avoid timelocks due to the stopped process $\mathbf{0}$, we replace it with the inactive process $\underline{\mathbf{0}}$ introduced in [9], which lets time pass according to the rule $\underline{\mathbf{0}} \xrightarrow{t} \underline{\mathbf{0}}$. To avoid timelocks caused by restriction, for both calculi we opt for the following two-level syntax featuring only restriction at the top level (let $\mathbb{P}'_{\rm TCCS}$ and $\mathbb{P}'_{\rm CIPA}$ be the two resulting sets of closed and guarded process terms):

$$\begin{array}{rcl} P' & ::= & P \mid P'\backslash L \\ P & ::= & \underline{\mathbf{0}} \mid \alpha.P \mid (t).P \mid P + P \mid P|P \mid P[\varphi] \mid X \mid \mathrm{rec}\, X : P \\ Q' & ::= & Q \mid Q'\backslash L \\ Q & ::= & \mathrm{nil} \mid a.Q \mid \mathrm{wait}\, t.Q \mid Q + Q \mid Q|Q \mid Q[\varphi] \mid X \mid \mathrm{rec}\, X : Q \end{array}$$

Note that this modification does not limit the expressive power of the calculi. Suppose that a process P is made out of three subprocesses P_1, P_2, P_3 composed in parallel, such that P_1 has action a and the other two have action $\bar{a}$, but only P_1 and P_2 have to synchronize on a and $\bar{a}$. Normally, one would write $(P_1|P_2)\backslash\{a\}|P_3$, but this is forbidden by the revised syntax. However, the same effect can be obtained through $(P_1[\varphi]|P_2[\varphi]|P_3)\backslash\{b\}$, where relabeling function φ maps a to a fresh action b not occurring in any of the three subprocesses.

3.2 From Delays to Durations

One of the major design decisions about the translation of (modified) TCCS into (modified) CIPA is how to assign durations to actions. In principle, it is desirable to be able to associate a suitable nonzero duration with every visible action occurring in a TCCS process that is not untimed. Unfortunately, in most cases this is not possible, as we now show.

Consider the TCCS process $a.(t_1).b.(t_2).\underline{\mathbf{0}}$. In this case, it is natural to interpret delay t_1 as the duration of a and delay t_2 as the duration of b, thus considering the occurrence of an instantaneous visible action of TCCS as the *beginning* of the corresponding durational action of CIPA (*initial view*). In the TCCS process $(t_1).a.(t_2).b.\underline{\mathbf{0}}$, the durations are as before, provided that the occurrence of an instantaneous visible action is considered as the *end* of the corresponding durational action (*final view*). Notice that, if $a = b$ but $t_1 \neq t_2$, the translation into a reasonably corresponding CIPA process would not be possible, unless, as noted in [5], we further extend the duration function for CIPA by admitting that different occurrences of the same action may have different durations.

Let us now examine the case in which there is not a precise pairing between actions and delays, like, e.g., in the TCCS process $(t_1).a.(t_2).\underline{\mathbf{0}}$. In this scenario, the duration of a can be either t_1 or t_2, but in any case a waiting is necessary to account for the delay that is not associated with a. The situation is even more complicated if we consider the TCCS process $a.(t_1).(t_2).b.\underline{\mathbf{0}}$. One option is to interpret $t_1 + t_2$ as the duration of a (initial view), with b having duration 0. The dual option is to interpret $t_1 + t_2$ as the duration of b (final view), with a having duration 0. In any case, the definition of the encoding would become technically involved, especially in the presence of recursion, due to the necessity of performing some lookahead. Moreover, there seems not to be any strong reason for choosing one option rather than the other.

Yet another option is to interpret t_1 as the duration of a (initial view) and t_2 as the duration of b (final view). This mixed option should be discarded because it disrupts equivalence preservation of the encoding. Indeed, the considered process is equivalent to the TCCS process $a.(t_2).(t_1).b.\underline{\mathbf{0}}$, while, under the assumption $t_1 \neq t_2$, the two corresponding CIPA processes are not equivalent to each other, because the a-transition of duration t_1 in the first CIPA process cannot be matched by the a-transition of duration t_2 of the second CIPA process.

Summing up, on the one hand there are TCCS delays that cannot be associated with any visible action, and hence have to be translated into CIPA

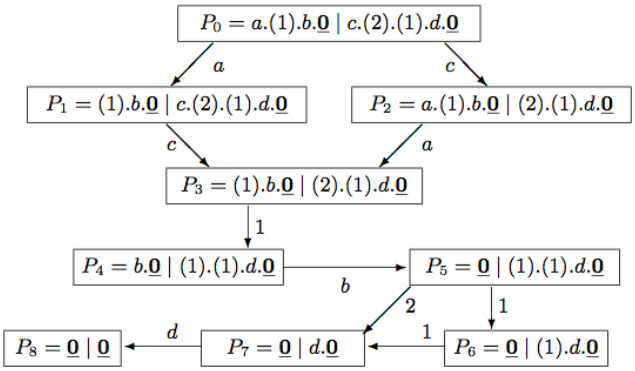

Fig. 1. Labeled transition system for P_0

waitings. On the other hand, it is not always possible to assign a nonzero duration to every TCCS visible action. In particular, this is impossible in the case of untimed TCCS processes. In this respect, it is also worth reminding that the two untimed TCCS processes $a.\underline{0}|b.\underline{0}$ and $a.b.\underline{0} + b.a.\underline{0}$ are always equivalent under the interleaving view of concurrency, while the two ideally corresponding CIPA processes $a.\mathrm{nil}|b.\mathrm{nil}$ and $a.b.\mathrm{nil} + b.a.\mathrm{nil}$ are equivalent to each other only if both a and b have duration zero.

As a consequence, for the sake of simplicity, uniformity, and semantics preservation, when encoding TCCS into CIPA we proceed as follows:

- Every TCCS action a will be translated into a CIPA action a with $\Delta(a) = 0$.
- The TCCS action τ will be translated into a CIPA waiting of duration 0 by allowing for waitings of the form $\mathrm{wait}\,0.Q$ in the modified syntax of CIPA.
- Every TCCS delay t will be translated into a CIPA waiting of duration t.

3.3 Which Behavioral Equivalence Can Be Preserved?

While the encoding from CIPA to TCCS defined in [5] preserves strong timed bisimilarity, this cannot be the case for the reverse encoding from (modified) TCCS to (modified) CIPA.

Consider the two TCCS processes $a.(t_1).(t_2).b.\underline{0}$ and $a.(t_1 + t_2).b.\underline{0}$, which are equivalent to each other according to the strong timed bisimilarity of [8]. Their corresponding CIPA processes will be respectively $a.\mathrm{wait}\,t_1.\mathrm{wait}\,t_2.b.\mathrm{nil}$ and $a.\mathrm{wait}\,(t_1+t_2).b.\mathrm{nil}$, which are not equivalent to each other according to the strong timed bisimilarity defined in [5].

It is however worth pointing out that the two former processes are equivalent according to $\approx_{\mathrm{TCCS}}$ and, most importantly, the two latter processes are equivalent according to $\approx_{\mathrm{CIPA}}$. As a consequence, in this paper we have to restrict ourselves to weak timed bisimilarities when investigating semantics preservation for the reverse encoding.

4 The Reverse Encoding

In this section, we translate (modified) TCCS process terms into (modified) CIPA process terms and we show that the resulting encoding is fully abstract, in the sense that it preserves weak timed bisimilarity. The encoding is defined by induction on the syntactical structure of process terms.

Definition 5. *The encoding* $[\![\_]\!] : \mathbb{P}'_{\rm TCCS} \to \mathbb{P}'_{\rm CIPA}$ *is defined as follows:*

$$\begin{array}{rcl@{\qquad}rcl} [\![\underline{\mathbf{0}}]\!] & = & \mathrm{nil} & [\![a.P]\!] & = & a.[\![P]\!] \\ [\![\tau.P]\!] & = & \mathrm{wait}\,0.[\![P]\!] & [\![(t).P]\!] & = & \mathrm{wait}\,t.[\![P]\!] \\ [\![P_1 + P_2]\!] & = & [\![P_1]\!] + [\![P_2]\!] & [\![P_1|P_2]\!] & = & [\![P_1]\!]|[\![P_2]\!] \\ [\![P\backslash L]\!] & = & [\![P]\!]\backslash L & [\![P[\varphi]]\!] & = & [\![P]\!][\varphi] \\ [\![X]\!] & = & X & [\![rec\,X : P]\!] & = & rec\,X : [\![P]\!] \end{array}$$

with $\Delta(a) = 0$ *for all* $a \in Act \setminus \{\tau\}$. ■

The states of the labeled transition systems underlying a TCCS process P and the corresponding $[\![P]\!]$ are strictly related. This relation is the key point that permits to show the main result of the paper stated in the forthcoming Thm. 1. Formally, to establish the relation, we need to add local clocks to $[\![P]\!]$. We let $\mathcal{E}[\![P]\!] \in \mathbb{KP}$ denote the encoded P process where a clock $0 \Rightarrow$ has been added to each sequential component.

Consider as an example the process $P_0 = a.(1).b.\underline{\mathbf{0}} \mid c.(2).(1).d.\underline{\mathbf{0}}$, whose transition system is depicted in Fig. 1. The transition system of $\mathcal{E}[\![P_0]\!]$ is shown in Fig. 2. It easy to see that, as far as only visible actions and zero-valued local clocks are concerned, the correspondence is one-to-one: $\mathcal{E}[\![P_i]\!] = K_i$, for $i = 0, 1, 2, 3$. However, in CIPA the wait t waitings, which produce τ actions, can be executed independently by one of the sequential components, which moves its own clock forward in time. In the meanwhile, the other components still have to execute actions "before" that time. This happens, for instance, in $K_1 \xrightarrow[1]{\tau@0} K'_1$; note, anyway, that K'_1 can still perform the visible action c at time 0.

Processes P_5, P_6 and P_7 are related by TCCS delay transitions. In the corresponding CIPA states, we can relate P_5 with K_5. The nil CIPA process lets any time pass for other sequential components. Thus, K_5 can be considered equivalent (at least with respect to $\approx_{\rm CIPA}$) to $K'_5 = 2 \Rightarrow \mathrm{nil} \mid 2 \Rightarrow \mathrm{wait}\,1.d.\mathrm{nil}$. In this way, $\mathcal{E}[\![P_6]\!] = 0 \Rightarrow \mathrm{nil} \mid 0 \Rightarrow \mathrm{wait}\,1.d.\mathrm{nil}$ can be related to K'_5 by adjusting the clock values that are different in absolute value, but agree on the relative differences: $2 - 2 = 0 - 0 = 0$. Similarly, process P_7 can be related to process $K'_6 = 3 \Rightarrow \mathrm{nil} \mid 3 \Rightarrow d.\mathrm{nil}$ obtained from K_6.

Consider, finally, process P_4, derived from P_0. Its counterpart K_4 cannot perform any τ, but $\mathcal{E}[\![P_4]\!]$ has clock values not corresponding to K_4. Nevertheless, the b action is performed at the same time in both cases, i.e., with timestamp 1.

To formally treat the discrepancies mentioned above, it is convenient to define a structural congruence $\equiv$ over $\mathbb{KP}$ that permits to equate timed processes that respect, in their sequential components, the following two equations:

$$\begin{array}{rcl} m \Rightarrow \mathrm{wait}\,n.P & = & m + n \Rightarrow P \\ n \Rightarrow \mathrm{nil} & = & m \Rightarrow \mathrm{nil} \end{array}$$

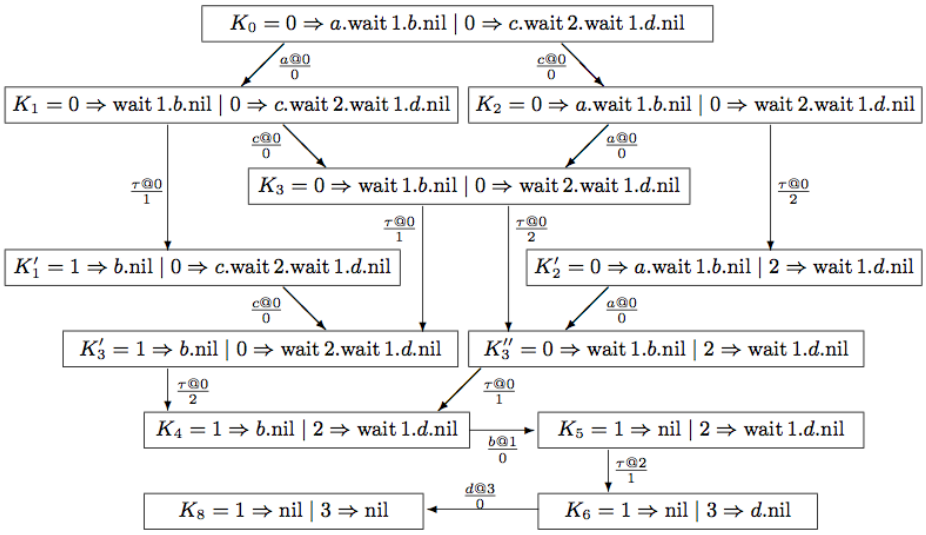

Fig. 2. Labeled transition System for $\mathcal{E}[\![P_0]\!]$

where $n, m \in \mathbb{N}$. It is easy to see that $\equiv$ implies $\approx_{\text{CIPA}}$. Moreover, following [5], we define $wf \subseteq \mathbb{KP} \times \mathbb{N}$ and $up\colon \mathbb{KP} \times \mathbb{N} \to \mathbb{KP}$. We let $wf(K, n)$ hold iff the local clocks of K can be decreased by n without any of them becoming negative, neglecting the components that are nil. If $wf(K, n)$, then $up(K, n)$ is the timed CIPA state K in which every local clock (apart from the nil components) has been decreased by n.

Using this notation, we get $\mathcal{E}[\![P_1]\!] = K_1 = up(K_1, 0) \equiv K'_1$, i.e., P_1 in Fig. 1 can be related to two timed states that are structural equivalent and, thus, $\approx_{\text{CIPA}}$ equivalent. Moreover, $\mathcal{E}[\![P_5]\!] = up(K_5, 1)$, $\mathcal{E}[\![P_6]\!] = up(K'_5, 1 + 1 = 2)$ where $K'_5 \equiv K_5$, $\mathcal{E}[\![P_7]\!] = up(K'_6, 2 + 1 = 3)$ where $K'_6 \equiv K_6$. Note that the subsequent times n in $up(\cdot, n)$ reflect the TCCS delay transitions. Finally, $\mathcal{E}[\![P_4]\!] = 0 \Rightarrow b.\text{nil} \mid 0 \Rightarrow \text{wait } 1.\text{wait } 1.d.\text{nil} \equiv 0 \Rightarrow b.\text{nil} \mid 1 \Rightarrow \text{wait } 1.d.\text{nil} = up(K_4, 1)$.

Theorem 1. *Let $P_1, P_2 \in \mathbb{P}'_{\text{TCCS}}$. Then $P_1 \approx_{\text{TCCS}} P_2$ iff $[\![P_1]\!] \approx_{\text{CIPA}} [\![P_2]\!]$.* ■

5 Conclusions

In this paper, we have addressed the issues raised at the end of [5] and shown that it is possible, after applying certain modifications to the languages, to define a reverse semantics-preserving mapping from TCCS to CIPA. Unlike the direct encoding of [5] from CIPA to TCCS, which preserves strong timed bisimilarity, our reverse encoding can only preserve weak timed bisimilarity.

As future work, we want to investigate how to exploit our encoding of TCCS into CIPA together with the notion of compact representation of CIPA timed states introduced in [5], to achieve a better performance with respect to weak

timed bisimilarity checking algorithms for TCCS. Moreover, we would like to extend the reverse mapping so to encode the idling operator and the weak choice operator of TCCS as well. Similar to [5], we also plan to investigate variants of our reverse encoding in which laziness or maximal progress is assumed in place of action urgency. Finally, continuing the work of [3], we would like to provide a uniform framework for comparing the various timed process calculi and timed models that have been proposed in the literature.

References

1. L. Aceto and D. Murphy. Timing and causality in process algebra. *Acta Informatica*, 33:317–350, 1996.
2. J. Baeten and J. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1991.
3. M. Bernardo and L. Tesei. Encoding timed models as uniform labeled transition systems. In *Proc. of the 10th European Performance Engineering Workshop (EPEW 2013)*, volume 8168 of *LNCS*, pages 104–118. Springer, 2013.
4. T. Bolognesi and F. Lucidi. LOTOS-like process algebras with urgent or timed interactions. In *Proc. of the 4th Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE 1991)*, volume C-2 of *IFIP Transactions*, pages 249–264, 1991.
5. F. Corradini. Absolute versus relative time in process algebras. *Information and Computation*, 156:122–172, 2000.
6. F. Corradini, W. Vogler, and L. Jenner. Comparing the worst-case efficiency of asynchronous systems with PAFAS. *Acta Informatica*, 38:735–792, 2002.
7. M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.
8. F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proc. of the 1st Int. Conf. on Concurrency Theory (CONCUR 1990)*, volume 458 of *LNCS*, pages 401–415. Springer, 1990.
9. F. Moller and C. Tofts. Behavioural abstraction in TCCS. In *Proc. of the 19th Int. Coll. on Automata, Languages and Programming (ICALP 1992)*, volume 623 of *LNCS*, pages 559–570. Springer, 1992.
10. X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Proc. of the REX Workshop on Real Time: Theory in Practice*, volume 600 of *LNCS*, pages 526–548. Springer, 1991.
11. X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114:131–178, 1994.
12. J. Quemada, D. de Frutos, and A. Azcorra. TIC: A timed calculus. *Formal Aspects of Computing*, 5:224–252, 1993.
13. G. Reed and A. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.
14. I. Ulidowski and S. Yuen. Extending process languages with time. In *Proc. of the 6th Int. Conf. on Algebraic Methodology and Software Technology (AMAST 1997)*, volume 1349 of *LNCS*, pages 524–538. Springer, 1997.
15. W. Yi. CCS + time = an interleaving model for real time systems. In *Proc. of the 18th Int. Coll. on Automata, Languages and Programming (ICALP 1991)*, volume 510 of *LNCS*, pages 217–228. Springer, 1991.