# Time Abstracted Bisimulation: Implicit Specifications and Decidability

Kim G. Larsen*and Wang Yi†

**Abstract**

In the last few years a number of real–time process calculi have emerged with the purpose of capturing important quantitative aspects of real–time systems. In addition, a number of process equivalences sensitive to time–quantities have been proposed, among these the notion of timed (bisimulation) equivalence in [RR86, DS89, HR91, BB89, NRSV90, MT90, Wan91b].

In this paper, we introduce a *time–abstracting* (bisimulation) equivalence, and investigate its properties with respect to the real–time process calculus of [Wan90]. Seemingly, such an equivalence would yield very little information (if any) about the timing properties of a process. However, time–abstracted reasoning about a composite process may yield important information about the relative timing–properties of the components of the system. In fact, we show as a main theorem that such implicit reasoning will reveal *all* timing aspects of a process. More precisely, we prove that two processes are interchangeable in any context up to time–abstracted equivalence precisely if the two processes are themselves timed equivalent.

As our second main theorem, we prove that time–abstracted equivalence is decidable for the calculus of [Wan90] using classical methods based on a finite–state symbolic, structured operational semantics.

## 1 Introduction

During the last few years various process calculi have been extended to include real–time in order to handle quantitative aspects of real–time systems, for instance that some critical event must not or should happen within a certain time period. The extensions often include timed versions of classical process equivalences, e.g. timed bisimulation equivalence, timed failure equivalence and timed trace equivalence [RR86, DS89, HR91, NRSV90, MT90, Wan91b]. Loosely speaking, for two processes to be equivalent they should not only agree on *what* actions they can perform, they must also agree on *when* these actions are performable. Alternatively, one can say that an observer is assumed to be sensitive to passage of time including the quantity by which time is passing.

A fundamental problem induced by any new process calculus is that of axiomatization and decidability of the associated process equivalence. Normally, these problems are solved in two stages: the problems are first solved for the class of regular processes, i.e. processes with no parallel composition, after which it is shown how to remove parallel composition through the use of a so–called expansion theorem. However, for real–time calculi where time is represented

*Department of Mathematics and Computer Science, Institute for Electronic Systems, Aalborg University, DK9220 Aalborg, Denmark. E-mail: kgl@iesd.auc.dk. The work of this author was supported by the Danish Natural Science Research Council project DART and partially by the ESPRIT Basic Research Action 7166, CONCUR2.

†Department of Computer Systems, Box 325, Uppsala University, S751 05, Uppsala, Sweden. E-mail: yi@docs.uu.se. The work of this author was done during a visit to the deparment of mathematics and computer science at Aalborg University, and supported by the Danish Natural Science Research Council project DART and partially by the ESPRIT Basic Research Action, REACT.

by some dense time domain (such as the non-negative reals) processes will have infinitely many states, and it has been shown in [GL92] that no expansion theorem exists for timed bisimulation equivalence — i.e. parallel composition can not in general be removed. This explains why axiomatization and decidability of various equivalences between real-time processes based on dense time domains have proven notoriously hard problems. Recent work by Čerāns [Č92], Chen [Che91b] and Fokkink and Klusener [FK91] offers the first examples of decidability and axiomatization for real-time calculi based on dense time.

In this paper we introduce a *time-abstracting* (bisimulation) equivalence between real-time processes, i.e. in comparing real-time processes we shall abstract away from passage of time [1]. Seemingly, such an equivalence would yield very little information (if any at all) about the timing behaviour of a real-time system. However, if the real-time system is a combination of real-time systems, $O(P_1, \ldots, P_n)$ say, time-abstracted reasoning will at least yield some information about the relationship between the concrete timing properties of the components $P_1, \ldots, P_n$. In fact, as we shall prove as a main theorem of this paper, in a certain formal sense *all* timing aspects of a real-time system may be revealed in this manner.

As the second main contribution of this paper, we demonstrate that the time-abstracted equivalence is decidable using essentially classical methods based on a finite-state symbolic, structured operational semantics. The symbolic semantics is based on a discrete version of the standard (continuous) operational semantics. In order to obtain completeness it is essential that the symbolic semantics is based on a sufficiently fine "granularity". In fact, we show that the "granularity" required is linearly dependent on the number of parallel components.

To further motivate the usefulness of time-abstracted equivalence consider the combined system in Figure 1 consisting of two (disposable) media $A$ and $B$.

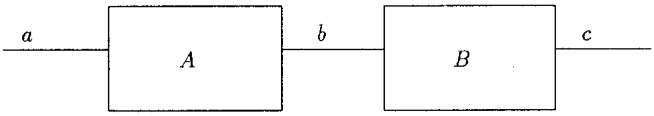


Figure 1: A Combined Medium

Functionally, the two media are nearly identical: they accept messages on the left port passing them on to the right port. However, taking time into account, there are important differences between the media: after having accepted a message on port $a$, $A$ is immediately able to deliver the massage on port $b$. However, if the message has not been taken after a delay of $t_a$ a timeout will occur and the massage is lost. In contrast, the medium $B$ will never lose a message once it has been accepted. However, a message can only be accepted on port $b$ after some initial delay $t_b$. Using the timed calculus of Wang [Wan90, Wan91b, Wan91a] the two media $A$ and $B$ may be specificed as follows:

$$\begin{aligned} A &\stackrel{def}{=} a.(\bar{b}.\mathrm{nil} + \epsilon(t_a).\tau.\mathrm{nil}) \\ B &\stackrel{def}{=} \epsilon(t_b).b.\bar{c}.\mathrm{nil} \end{aligned}$$

It should be obvious that even from a time-abstracted point of view, the behaviour of the combined system $(A \mid B)\backslash b$ is highly dependent on the timing parameters $t_a$ and $t_b$. Essentially, if $t_a > t_b$ the combined system will function as a proper (disposable) medium, i.e.:

$$(A \mid B)\backslash b \stackrel{\bullet}{\approx} a.\bar{c}.\mathrm{nil} \tag{1}$$

[1]This abstraction is very similar to the abstraction from internal computation in classical process algebras.

where $\stackrel{\bullet}{\approx}$ denotes our (weak) time–abstracting equivalence [2]. In contrast, if $t_b > t_a$, the combined medium may not be able to successfully deliver messages; in fact the following will hold [3]:

$$(A \mid B)\backslash b \stackrel{\bullet}{\approx} a.(\tau.\mathrm{nil} + \tau.\bar{c}.\mathrm{nil}) + \tau.a.\bar{c}.\mathrm{nil} \tag{2}$$

Even though, we gain information about the *relationship* of the timing behaviours of $A$ and $B$ in both (1) and (2), we have no information about the timing behaviour of the combined system. Obviously, in the case (1) a message can be delivered on port $c$ after a delay of less than $t_b$ from the acceptance of the message. Using the (weak) timed bisimulation equivalence from [Wan91a] such properties can be specified:

$$(A \mid B)\backslash b \approx a.\epsilon(t_b).\bar{c}.\mathrm{nil}^4$$

Alternatively, one can express such *explicit* timing properties using Timed Modal Logics, e.g. [ACD90, HLW91, HNJ92, RH92]. However, we can also formulate explicit timing properties using time–abstracted equivalence by resorting to *implicit specifications*; i.e. instead of specifying properties of $S = (A \mid B)\backslash b$ directly we specify properties of the system $S$ in certain *contexts*. Concretely, specifying that $S$ must be able to deliver on port $c$ after a delay of no more than $d$ after acceptance on port $c$ can be expressed as follows:

$$(\bar{a}.(c.w.\mathrm{nil} + \epsilon(d).\tau.\mathrm{nil}) \mid S)\backslash\{a, c\} \quad \stackrel{\bullet}{\approx} \quad w.\mathrm{nil} \tag{3}$$

where $w$ is a distinguished (success) action. Here, we are exploiting the *maximal progress* property of the calculus in [Wan91a] [5].

The previously announced main theorem, that all explicit timing properties can be captured using time–abstracted equivalence, can now be made more precise: we show that implicit time–abstracting specifications of the form (3) precisely characterizes *timed* bisimulation equivalence. That is, two timed processes are timed bisimulation equivalent just in case they satisfy the same implicit time–abstracted specifications. Thus, without any loss of discriminating power, one may use time–abstracting bisimulation equivalence instead of timed bisimulation equivalence.

The outline of the paper is as follows: in section 2 we review the timed calculus of [Wan90, Wan91b, Wan91a] together with the notion of timed bisimulation; in section 3 strong and weak notions of time–abstracted bisimulations are introduced; in section 4 we prove as our first main theorem that implicit time–abstracting specifications are as discriminating as timed bisimulation; section 5 contains our second main contribution: decidability of strong and weak time–abstracted bisimulation equivalence. Finally, in section 6 we give some concluding remarks. To achieve readability while maintaining credibility we enclose full proofs in the appendices.

# 2 Timed Processes

## 2.1 Syntax and Semantics

The language we use to describe timed processes is essentially, Milner's CCS extended with a delay construct $\epsilon(d).P$. Informally, $\epsilon(d).P$ means "wait for $d$ units of time and then behave like $P$", where $d \in \mathcal{R}_+$ is a nonnegative real.

---

[2] Weak indicating that $\stackrel{\bullet}{\approx}$ also abstracts from internal computation.

[3] The summand $\cdots \tau.a.\bar{c}.\mathrm{nil}$ reflects that messages *may* successfully be delivered in case $A$ delays sufficiently long before accepting a messages as this will reduce the remaining delay for $B$.

[4] The displayed equivalence does in fact not hold as the delay required before the delivery depends on the delay before the acceptance. Using time–variables as in [Wan91b] a valid equation would be: $(A \mid B)\backslash b \approx a@t.\epsilon(t_b - t).\bar{c}.\mathrm{nil}$

[5] Maximal progress means that time is not allowed to pass if a system can perform internal computation.

As in CCS, we assume a set $\Lambda = \Delta \cup \bar{\Delta}$ with $\bar{\bar{\alpha}} = \alpha$ for all $\alpha \in \Lambda$, ranged over by $\alpha, \beta$ representing external actions, and a distinct symbol $\tau$ representing internal actions. We use $\mathcal{A}ct$ to denote the set $\Lambda \cup \{\tau\}$ ranged over by $a, b$ representing both internal and external actions.

Further, assume a set of process variables ranged over by $X$.

We adopt a two-phase syntax to describe networks of regular timed processes. First, regular timed process expressions are generated by the following grammar:

$$E ::= \mathrm{nil} \mid X \mid \epsilon(d).E \mid a.E \mid E + E \mid X \stackrel{def}{=} E$$

We shall restrict process expressions to be *well-guarded* in the following sense:

**Definition 1** *$X$ is well-guarded in $E$ if and only if every free occurrence of $X$ in $E$ is within a subexpression (a guard) of the form $a.F$ in $E$.*

*$E$ is well-guarded if and only if every free variable in $E$ is well-guarded in $E$, and for every subexpression of the form $X \stackrel{def}{=} F$ in $E$, $X$ is well-guarded in $F$.* □

Closed and well-guarded expressions generated by the grammar above are called *regular timed processes*. Networks of regular timed processes are described by CCS parallel composition:

$$P_1|...|P_n$$

where $P_i$ are regular timed processes. For simplicity, we have ignored the other CCS operators. However, the results of this paper can be easily extended to more general types of networks modelled by the combination of parallel composition, restriction and relabelling:

$$(P_1[S_1]|...|P_n[S_n])\backslash A$$

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{\epsilon(0).P \xrightarrow{a} P'}$$

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'} \qquad \frac{P \xrightarrow{a} P'}{X \xrightarrow{a} P'}\ [X \stackrel{def}{=} P]$$

$$\frac{Q \xrightarrow{a} Q'}{P|Q \xrightarrow{a} P|Q'} \qquad \frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{a} P'|Q} \qquad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\bar{\alpha}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

Table 1: Action Rules for Timed Semantics.

$$\frac{}{\mathrm{nil} \xrightarrow{\epsilon(d)} \mathrm{nil}} \qquad \frac{}{\epsilon(c+d).P \xrightarrow{\epsilon(d)} \epsilon(c).P} \qquad \frac{P \xrightarrow{\epsilon(d)} P'}{\epsilon(c).P \xrightarrow{\epsilon(c+d)} P'}$$

$$\frac{}{\alpha.P \xrightarrow{\epsilon(d)} \alpha.P} \qquad \frac{P \xrightarrow{\epsilon(d)} P' \quad Q \xrightarrow{\epsilon(d)} Q'}{P + Q \xrightarrow{\epsilon(d)} P' + Q'} \qquad \frac{P \xrightarrow{\epsilon(d)} P'}{X \xrightarrow{\epsilon(d)} P'}\ [X \stackrel{def}{=} P]$$

$$\frac{P \xrightarrow{\epsilon(d)} P' \quad Q \xrightarrow{\epsilon(d)} Q'}{P|Q \xrightarrow{\epsilon(d)} P'|Q'}\ [Sort_d(P) \cap \overline{Sort_d(Q)} = \emptyset]$$

Table 2: Delay Rules for Timed Semantics .

We will use $P, Q$ to range over timed processes.

A timed operational semantics for the language has been developed in [Wan90]. We present the transition rules in two groups: rules for actions in table 1 [6] and rules for delays in table 2 [7].

Note that the side condition for the delay rule of parallel composition is to guarantee that the parallel processes satisfy the maximal progress assumption, that is, *a timed process will never wait if it can perform an internal action* $\tau$. The condition is formalized by means of $Sort_d(P)$ defined inductively on the structure of processes $P$, in table 3. Intuitively, $Sort_d(P)$ includes all external actions that $P$ is able to perform within $d$ time units; whereas $Sort_d(P) \cap \overline{Sort_d(Q)} = \emptyset$ [8] means that $P$ and $Q$ cannot communicate with each other within $d$ time units.

**Definition 2** *Given a process $P$, we define $Sort_0(P) = \emptyset$ and $Sort_c(P)$ for $c \neq 0$ to be the least set satisfying the equations [9] given in table 3.* □

$$
\begin{array}{rcl}
Sort_c(\mathrm{nil}) & = & \emptyset \\
Sort_c(\alpha.P) & = & \{\alpha\} \\
Sort_c(\tau.P) & = & \emptyset \\
Sort_c(\epsilon(d).P) & = & Sort_{c \dot{-} d}(P) \\
Sort_c(P+Q) & = & Sort_c(P) \cup Sort_c(Q) \\
Sort_c(X) & = & Sort_c(P) \quad [X \stackrel{def}{=} P] \\
Sort_c(P|Q) & = & Sort_c(P) \cup Sort_c(Q)
\end{array}
$$

Table 3: Equations for $Sort_c(P)$ .

The following properties of timed processes will be often referred in the later sections.

**Proposition 1**

1. *(maximal progress) If $P \xrightarrow{\tau} P'$ for some $P'$, then $P \xrightarrow{\epsilon(d)} P''$ for no $d$ and $P''$.*
2. *(time determinism) Whenever $P \xrightarrow{\epsilon(d)} P'$ and $P \xrightarrow{\epsilon(d)} P''$ then $P' = P''$.*
3. *(persistency) If $P \xrightarrow{\epsilon(d)} P'$ and $P \xrightarrow{\alpha} Q$ for some $P'$ and $Q$, then $P' \xrightarrow{\alpha} Q'$ for some $Q'$.*
4. *(time continuity) For all $c, d$ and $P''$, $P \xrightarrow{\epsilon(c+d)} P''$ iff $P \xrightarrow{\epsilon(c)} P' \xrightarrow{\epsilon(d)} P''$ for some $P'$.* □

We end this section with notation:

- $\overline{P}$ stands for a network $P_1|...|P_n$ where $P_i$ are regular timed processes.
- Whenever $P \xrightarrow{\epsilon(d)} P'$, $P^d$ stands for $P'$ [10] ; note that $P^d$ is well-defined due to time-determinism property stated above.
- $\overline{P}^{\overline{x}}$ stands for $P_1^{x_1}|...|P_n^{x_n}$ for $\overline{x} = (x_1, ..., x_n)$.

[6] Note that apart from the rule for $\epsilon(0).P$, the action rules are exactly the same as in CCS.

[7] In table 2, we use $d$ to stand for a non-zero real; this implies that a $\epsilon(0)$–transition can never be inferred by the inference rules. However, we shall apply the convention that $P \xrightarrow{\epsilon(0)} P$ for all $P$.

[8] Here, $\overline{Sort_d(Q)}$ is defined to be the set $\{\bar{\alpha} \mid \alpha \in Sort_d(Q)\}$.

[9] In table 3, $c \dot{-} d$ is defined to be $c - d$ if $c > d$, 0 otherwise.

[10] Note that $P^0$ stands for $P$ following the convention that $P \xrightarrow{\epsilon(0)} P$ for all $P$.

Conceptually, one can imagine each component $P_i$ of a network $\overline{P}$ to be equipped with a private clock. All clocks proceed at the same speed and a clock-value will be reset to 0 when the corresponding component perform a real action; $\overline{P}^{\overline{x}}$ denotes the state of $\overline{P}$ in which the clock-values are $x_1, ..., x_n$.

## 2.2 Timed Bisimulation

We have developed a labelled transition system: $\langle \mathcal{P}_R, \longrightarrow, \mathcal{L} \rangle$ where $\mathcal{P}_R$ is the set of timed processes generated by the two-phase syntax; $\longrightarrow$ is the least relation satisfying the inference rules given in table 1 and table 2; $\mathcal{L}$ is the set of labels, $\mathcal{A}ct \cup \{\epsilon(d) \mid d \in \mathcal{R}_+\}$. To compare timed processes, strong and weak notions of timed bisimulation have been defined based on this transition system in [Wan90].

**Definition 3** *(strong timed bisimulation) A binary relation $\mathcal{S}$ on $\mathcal{P}_R$ is a strong timed simulation if $(P, Q) \in \mathcal{S}$ implies that for all $a \in \mathcal{A}ct$ and $d \in \mathcal{R}_+$,*

1. *Whenever $P \xrightarrow{a} P'$ then, for some $Q'$, $Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathcal{S}$*
2. *Whenever $P \xrightarrow{\epsilon(d)} P'$ then, for some $Q'$, $Q \xrightarrow{\epsilon(d)} Q'$ and $(P', Q') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a strong timed bisimulation if it is symmetrical. The largest strong timed bisimulation is called strong timed equivalence, denoted $\sim$.* □

Weak timed equivalence is defined by abstracting away from internal actions.

**Definition 4**

1. *$P \overset{\tau}{\Longrightarrow} Q$ if $P(\xrightarrow{\tau})^* Q$*
2. *$P \overset{\alpha}{\Longrightarrow} Q$ if $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* Q$*
3. *$P \overset{\epsilon(d)}{\Longrightarrow} Q$ if $P(\xrightarrow{\tau})^* \xrightarrow{\epsilon(d_1)} (\xrightarrow{\tau})^* ... (\xrightarrow{\tau})^* \xrightarrow{\epsilon(d_n)} (\xrightarrow{\tau})^* Q$ where $d = \sum_{i \le n} d_i$.* □

**Definition 5** *(weak timed bisimulation) A binary relation $\mathcal{S}$ on $\mathcal{P}_R$ is a weak timed simulation if $(P, Q) \in \mathcal{S}$ implies that for all $a \in \mathcal{A}ct$ and $d \in \mathcal{R}_+$,*

1. *Whenever $P \xrightarrow{a} P'$ then, for some $Q'$, $Q \overset{a}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{S}$*
2. *Whenever $P \xrightarrow{\epsilon(d)} P'$ then, for some $Q'$, $Q \overset{\epsilon(d)}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a weak timed bisimulation if it is symmetrical. The largest weak timed bisimulation is called weak timed equivalence, denoted $\approx$.* □

In [Wan91a], it has been shown that $\sim$ is a congruence w.r.t all CCS operators and $\approx$ is a congruence w.r.t. all the other operators except summation and recursion.

# 3 Time Abstracted Equivalences

In analyzing a large system, we often need to make proper abstractions according to what properties of the system we are interested. One such example is weak timed equivalence, which abstracts away from internal actions. In this section, we develop notions of bisimulation abstracting away from both time delays and internal actions.

**Definition 6** *(abstracting away from time)*

1. $P \overset{\varepsilon}{\longrightarrow\!\!\bullet} Q$ *if* $P(\xrightarrow{\epsilon(d)})^*Q$
2. $P \overset{a}{\longrightarrow\!\!\bullet} Q$ *if* $P \overset{\varepsilon}{\longrightarrow\!\!\bullet}\xrightarrow{a}\overset{\varepsilon}{\longrightarrow\!\!\bullet} Q$ □

For example, $\epsilon(2).\alpha.P \overset{\varepsilon}{\longrightarrow\!\!\bullet} \epsilon(0.3).\alpha.P$, ..., $\epsilon(2).\alpha.P \overset{\varepsilon}{\longrightarrow\!\!\bullet} \alpha.P$. Here, we simply consider a timed transition like $P \xrightarrow{\epsilon(d)} Q$ as an empty transition $P \overset{\varepsilon}{\longrightarrow\!\!\bullet} Q$ where the quantitative part i.e. $d$ of the transition is ignored. This assumes that the observer (or environment) who makes the observation is *insensitive to time–quantities.* Naturally, we may identify two processes if they can not be distinguished by any time insensitive environment.

**Definition 7** *(strong time abstracted equivalence) A binary relation $\mathcal{S}$ on $\mathcal{P}_R$ is a strong time abstracted simulation if $(P,Q) \in \mathcal{S}$ implies that for all $a \in \mathcal{A}ct$ and $d \in \mathcal{R}_+$,*

1. *Whenever $P \xrightarrow{a} P'$ then, for some $Q'$, $Q \overset{a}{\longrightarrow\!\!\bullet} Q'$ and $(P',Q') \in \mathcal{S}$*
2. *Whenever $P \xrightarrow{\epsilon(d)} P'$ then, for some $Q'$, $Q \overset{\varepsilon}{\longrightarrow\!\!\bullet} Q'$ and $(P',Q') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a strong time abstracted bisimulation if it is symmetrical. The largest strong time abstracted bisimulation is called strong time abstracted equivalence, denoted $\dot{\sim}$.* □

For example, $\epsilon(2).\tau.\mathrm{nil}|\epsilon(1).\beta.\mathrm{nil} \;\dot{\sim}\; \tau.\mathrm{nil}|\beta.\mathrm{nil} \;\dot{\sim}\; \tau.\beta.\mathrm{nil} + \beta.\tau.\mathrm{nil}$. Note that in terms of timed bisimulation equivalence $\sim$, there is no regular process equivalent to the parallel process.

We make a further abstraction to abstract away from internal actions.

**Definition 8** *(abstracting away from time and $\tau$)*

1. $P \overset{\varepsilon}{\Longrightarrow\!\!\bullet} Q$ *if* $P(\xrightarrow{\epsilon(d)} \cup \xrightarrow{\tau})^*Q$
2. $P \overset{\alpha}{\Longrightarrow\!\!\bullet} Q$ *if* $P \overset{\varepsilon}{\Longrightarrow\!\!\bullet}\xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow\!\!\bullet} Q$ □

**Definition 9** *(weak time abstracted equivalence) A binary relation $\mathcal{S}$ on $\mathcal{P}_R$ is a weak time abstracted simulation if $(P,Q) \in \mathcal{S}$ implies that for all $\alpha \in \mathcal{A}ct - \{\tau\}$ and $\theta \in \{\tau\} \cup \{\epsilon(d) \mid d \in \mathcal{R}_+\}$,*

1. *Whenever $P \xrightarrow{\alpha} P'$ then, for some $Q'$, $Q \overset{\alpha}{\Longrightarrow\!\!\bullet} Q'$ and $(P',Q') \in \mathcal{S}$*
2. *Whenever $P \xrightarrow{\theta} P'$ then, for some $Q'$, $Q \overset{\varepsilon}{\Longrightarrow\!\!\bullet} Q'$ and $(P',Q') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a weak time abstracted bisimulation if it is symmetrical. The largest weak time abstracted bisimulation is called weak time abstracted equivalence, denoted $\dot{\approx}$.* □

Now, we can further simplify our example process $\epsilon(2).\tau.\mathrm{nil}|\epsilon(1).\beta.\mathrm{nil}$ to $\beta.\mathrm{nil}$ by the equation: $\epsilon(2).\tau.\mathrm{nil}|\epsilon(1).\beta.\mathrm{nil} \;\dot{\approx}\; \beta.\mathrm{nil}$.

It seems that every timed process would be time–abstracted equivalent to an untimed process which contains no delay–construct. This is not true for $\dot{\sim}$. For instance,

$$(\epsilon(1).\alpha.\mathrm{nil}|\beta.(\tau.\mathrm{nil} + \bar{\alpha}.\omega.\mathrm{nil}))\backslash\{\alpha\} \;\dot{\not\sim}\; P_{ccs}$$

$$\begin{tikzcd}
& \dot{\approx} & \\
\dot{\sim} \arrow[ur] & & \approx \arrow[ul] \\
& \sim \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 2: Ordering Timed and Time–Abstracted Equivalences with Strength.

for all untimed processes $P_{ccs}$. However, it is true for weak time abstracted equivalence that for each timed process $P$, there will be an untimed process $P_{ccs}$ such that $P \dot{\approx} P_{ccs}$. For instance, it is easy to prove $(\epsilon(1).\alpha.\text{nil}|\beta.(\tau.\text{nil} + \bar{\alpha}.\omega.\text{nil}))\backslash\{\alpha\} \dot{\approx} (\tau.\alpha.\text{nil}|\beta.(\tau.\text{nil} + \bar{\alpha}.\omega.\text{nil}))\backslash\{\alpha\}$.

We conclude this section with the commuting diagram shown in figure 2, which illustrates the relationship between *timed* and *time–abstracted* equivalences. The arrow in the diagram should be understood as set inclusion, that is: $\sim \subseteq \dot{\sim} \subseteq \dot{\approx}$ and $\sim \subseteq \approx \subseteq \dot{\approx}$. The proofs of these inclusions, that they are strict and also the only inclusions among the four equivalences are straightforward.

# 4 Implicit Time Abstraction

In this section we present our first main theorem: *two timed processes are strong (weak) timed equivalent if and only if they satisfy the same strong (weak) implicit time–abstracted specifications.* Here, a strong implicit time–abstracted specification of a process $P$ is an equation of the form:

$$A \,|\, P \;\dot{\sim}\; B \tag{4}$$

where $A$ and $B$ are real–time processes. That is $P \sim Q$ if and only if $P$ and $Q$ satisfy the same equations of the form (4). Alternatively, the results in this section say that $\sim$ $(\approx)$ is the coarsest equivalence contained in $\dot{\sim}$ $(\dot{\approx})$ which is preserved by the parallel composition of our calculus [11].

**Theorem 1** *$P \sim Q$ if and only if $P \,|\, N \dot{\sim} Q \,|\, N$ for all $N$.*

**Proof:** *Only If:* As $\sim$ is preserved by all operators of the calculus and since $\sim$ is contained in $\dot{\sim}$, it is obvious that this direction holds.

*If:* We show that the relation:

$$\mathcal{R} = \{(P, Q) \mid \text{ for all } N,\ P|N \dot{\sim} Q|N\}$$

is a strong timed bisimulation. Thus consider $(P, Q) \in \mathcal{R}$.

First consider an action–transition $P \xrightarrow{a} P'$ and let $\{Q_1, \ldots, Q_m\}$ be the set of all $a$–derivatives for $Q$ [12].

In case $m = 0$ (i.e. $Q$ has no $a$–transitions), $P \,|\, N \not\dot{\sim} Q \,|\, N$ for $N = \overline{a}.w.\text{nil} + \tau.\text{nil}$, where $w$ is a distinguished action not occurring in neither $P$ nor $Q$. However, this contradicts the assumption that $(P, Q) \in \mathcal{R}$.

[11] As $\sim$ is preserved by *all* operators of the calculus, $\sim$ is in fact the congruence induced by $\dot{\sim}$. This fact does not extend to the weak case, as $\approx$ is not — as usual — preserved by $+$.

[12] We use the easily established fact, that all processes definable in our calculus are *image–finite* in the sense that the set of derivatives under any action is finite.

Thus, $m > 0$. Now, assume that $(P', Q_i) \notin \mathcal{R}$ for all $i$. We shall show that this leads to a contradiction. However, under this assumption it follows from the definition of $\mathcal{R}$ that for each $i$ there exists a process $N_i$ such that $P' \mid N_i \not\dot{\sim} Q_i \mid N_i$. Now let:

$$N \stackrel{def}{=} \overline{a}.N' \qquad N' \stackrel{def}{=} \sum_{i=1}^{m} w_i.N_i + \tau.N'$$

where $w_i$ are distinct actions not occurring in neither $P$ nor $Q$. Note, that $N'$ is a time-stopped process (and $P \mid N'$ is time-stopped for any $P$) in the sense that no delay-transitions can take place. Now we claim that $P \mid N \not\dot{\sim} Q \mid N$ contradicting that $(P, Q) \in \mathcal{R}$. To argue for this consider the transition:

$$P \mid N \xrightarrow{\tau} P' \mid N' \tag{5}$$

A possible match for $Q \mid N$ must be of the form $Q \mid N \xrightarrow{\tau}\!\!\bullet\ R$. Due to the maximal progress property of our calculus, and as $N'$ (and hence $Q_i \mid N'$) is time-stopped, the only possible such transitions are either of the form (a) $Q \mid N \xrightarrow{\tau}\!\!\bullet\ Q_i \mid N'$ or of the form (b) $Q \mid N \xrightarrow{\tau}\!\!\bullet\ Q'' \mid N$ with $Q \xrightarrow{\tau}\xrightarrow{\epsilon}\!\!\bullet\ Q''$. Clearly, transitions of the form (b) can not match (5) as $P' \mid N' \xrightarrow{w_i}$ whereas $Q'' \mid N \not\xrightarrow{w_i}\!\!\bullet$. Let us thus compare behaviours of $P' \mid N'$ and $Q_i \mid N'$: first note that with respect to $w_i$ both possess the following unique transitions: $P' \mid N' \xrightarrow{w_i} P' \mid N_i$ and $Q_i \mid N' \xrightarrow{w_i} Q_i \mid N_i$. Thus, if $P' \mid N' \dot{\sim} Q_i \mid N'$ it follows that $w_i.(P' \mid N_i) \dot{\sim} w_i.(Q_i \mid N_i)$. However, this contradicts the assumption that $P' \mid N_i \not\dot{\sim} Q_i \mid N_i$ and the easily established fact that whenever $a.U \dot{\sim} a.V$ then also $U \dot{\sim} V$. Thus, $Q \mid N$ has no match for the transition (5) of $P \mid N$ and hence $P \mid N \not\dot{\sim} Q \mid N$ contradicting the assumption that $(P, Q) \in \mathcal{R}$.

Now consider a delay transition $P \xrightarrow{\epsilon(d)} P'$. If $Q \not\xrightarrow{\epsilon(d)}$ then clearly $P \mid N \not\dot{\sim} Q \mid N$ for $N = \epsilon(d).w.\text{nil}$ contradicting $(P, Q) \in \mathcal{R}$. Otherwise assume that $Q \xrightarrow{\epsilon(d)} Q'$ (due to time-determinism $Q'$ is unique). Assume $(P', Q') \notin \mathcal{R}$, that is $P' \mid N' \not\dot{\sim} Q' \mid N'$ for some $N'$. In this case $P \mid N \not\dot{\sim} Q \mid N$ for $N = \epsilon(d).N'$ again violating the basic assumption that $(P, Q) \in \mathcal{R}$. □

**Example.** Consider the two processes [13]:

$$P = \epsilon(1).a \mid b \qquad Q = b.\epsilon(1).a + \epsilon(1).(a \mid b)$$

It is obvious that these two processes are not strong timed equivalent, i.e. $P \not\sim Q$. To see this, note that $P$ possesses the following transition-sequence:

$$P \xrightarrow{\epsilon(.5)} \epsilon(.5).a \mid b \xrightarrow{b} \epsilon(.5).a \mid \text{nil} \xrightarrow{\epsilon(.5)} a \mid \text{nil}$$

The only possible match $Q$ for is the following:

$$Q \xrightarrow{\epsilon(.5)} b.\epsilon(1).a + \epsilon(.5).(a \mid b) \xrightarrow{b} \epsilon(1).a \xrightarrow{\epsilon(.5)} \epsilon(.5).a$$

However, it is clear that this is not a proper match as $a \mid \text{nil} \xrightarrow{a}$ whereas $\epsilon(.5).a \not\xrightarrow{a}$. Now using the construction of the above theorem 1 we obtain the following process:

$$N = \epsilon(.5).\overline{b}.w_1.\epsilon(.5).(\overline{a}.w + \tau.\text{nil})$$

which distinguishes $P$ and $Q$, i.e. $P \mid N \not\dot{\sim} Q \mid N$. □

We have a similar result for weak *timed* and *time abstracted* equivalences.

**Theorem 2** *$P \approx Q$ if and only if $P \mid N \dot{\approx} Q \mid N$ for all $N$.*

[13] We are using the convention of dropping trailing nil's. That is, we write simply $a$ for $a$.nil.

**Proof:** *Only if:* As $\approx$ is preserved by parallel composition and since $\approx$ is contained in $\stackrel{\bullet}{\approx}$, it is obvious that this direction holds.

*If:* We show that the relation: $\mathcal{R} = \{(P, Q) \mid \text{for all } N,\ P|N \stackrel{\bullet}{\approx} Q|N\}$ is a weak timed bisimulation. A complete proof is given in the full version of the paper [LW93]. $\square$

# 5 Decidability

From the delay rules in table 2, we can easily see that the timed processes are infinite-state w.r.t. $\longrightarrow$ and also $\longrightarrow\!\!\!\bullet$. For example, $\epsilon(1).P|Q \stackrel{\varepsilon}{\longrightarrow\!\!\!\bullet} \epsilon(0.3).P|Q \stackrel{\varepsilon}{\longrightarrow\!\!\!\bullet} \ldots \stackrel{\varepsilon}{\longrightarrow\!\!\!\bullet} \epsilon(0.0005).P|Q \stackrel{\varepsilon}{\longrightarrow\!\!\!\bullet} \ldots \stackrel{\varepsilon}{\longrightarrow\!\!\!\bullet} P|Q$. The infinite-stateness makes the decidability problem of $\sim$ and $\stackrel{\bullet}{\sim}$ notoriously hard.

To achieve decidability, we shall study a particular class of processes $\mathcal{P}_N$ ranged over by $\overline{P}, \overline{Q}$, called integer processes in which, only naturals are allowed to occur in a delay operator $\epsilon(d)$. However, we should point out that the decidability result is easily extended to processes using rational numbers in delay operators: before comparing two such processes simply multiply all delays with a common constant, sufficiently large to make all delays integers.

In this section, we prove that strong (weak) time abstracted equivalence over integer processes is decidable. The proof is constructed in two steps. First, we show that the state-space of a timed process can be partitioned into equivalence classes according to the notion of time region due to Alur and Dill, [AD90]. Secondly, we develop a time-step semantics called $k$-semantics which is parameterized with a granularity $1/k$. Intuitively, the $k$-semantics describes how a process shall behave in every $1/k$ time units. The idea is to use each state of such a time-step semantics to represent an equivalence class of states of the timed semantics. Based on the parameterized $k$-semantics we define a family of symbolic time abstracted equivalences $\stackrel{\circ}{\sim}_k$ which is also relativized to the granularity $1/k$. It turns out that $\stackrel{\circ}{\sim}_{n+2}$ coincides with $\stackrel{\bullet}{\sim}$, that is:

$$\overline{P} \stackrel{\bullet}{\sim} \overline{Q} \textit{ if and only if } \overline{P} \stackrel{\circ}{\sim}_{n+2} \overline{Q}$$

where $n$ is the maximal number of components in the networks $\overline{P}$ and $\overline{Q}$.

Since the integer processes in the $(n+2)$-semantics are finite-state, $\stackrel{\circ}{\sim}_{n+2}$ can be checked using the existing techniques and algorithms for bisimulation-checking, such as [KS90, SV89, PT87, JGZ89, CPS89] and hence so can $\stackrel{\bullet}{\sim}$. Finally, we extend the results to weak time abstracted equivalence.

## 5.1 Partitioning State–Space into Equivalence Classes

To illustrate the idea, we consider a simple regular process:

$$P \stackrel{def}{=} \alpha.Q + \epsilon(1).\tau.R$$

The process may offer $\alpha$ before 1 and will time out at 1. Indeed it is infinite-state since by performing an empty transition (delay) it may reach a continuum of states, $\{P^x | x < 1\}$. However, $P^x \stackrel{\bullet}{\sim} P^y$ for all $x, y < 1$, that is, $\{P^x | x < 1\}$ is an equivalence class.

Naturally, we may say that all time points such as $x = 0, 0.1, \ldots, 0.9$ in the region $x < 1$ are equivalent in the sense that they give rise to an equivalence class of states. This motivates a notion of equivalence over time points in a multi-dimensional time vector.

Let $\overline{x}$ and $\overline{y}$ range over $\mathcal{R}_+^n$, understood as time points in the $n$-dimensional time vector. For $\overline{x} \in \mathcal{R}_+^n$ and $d \in \mathcal{R}_+$, we shall write $\overline{x} + d$ for $(x_1 + d, \ldots, x_n + d)$.

**Definition 10** *$\overline{x}$ and $\overline{y}$ are equivalent, denoted by* $\overline{x} \doteq \overline{y}$ *if*

1. $\forall i : (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$,
2. $\forall i, j : (\{x_i\} \leq \{x_j\} \iff \{y_i\} \leq \{y_j\})$ *and*
3. $\forall i : (\{x_i\} = 0 \iff \{y_i\} = 0)$.

*where $\lfloor d \rfloor$ is the lower integer part of $d$ and $\{d\}$ is the fractional part of $d$. The equivalence classes of $\mathcal{R}_+^n$ are called time regions.* □

The definition above is the standard one for time region, taken from [AD90]. The first clause requires that the lower integer parts of $\overline{x}$ and $\overline{y}$ must be equal; the second clause requires that the fractional parts of $\overline{x}$ and $\overline{y}$ must be ordered in the same way; the third requires that some fractional parts of $\overline{x}$ are 0 if and only if the corresponding fractional parts of $\overline{y}$ are 0.

The following is an important property of $\doteq$, saying that equivalent points —which must be in the same region—can always reach the same regions by delays.

**Lemma 1** *Whenever $\overline{x} \doteq \overline{y}$, then for all $d \in \mathcal{R}_+$, $\overline{x} + d \doteq \overline{y} + e$ for some $e \in \mathcal{R}_+$.*

**Proof:** It is given in the full version of the paper [LW93]. □

We intend to establish that for any integer parallel process $\overline{P}$, a time region denotes an equivalence class of states $\overline{P}^{|\overline{x}|}$ [14] in terms of $\dot{\sim}$. Thus, two states in a time region should agree on what actions they can perform and then reach the same regions; they should also be able to reach the same regions by delays.

**Lemma 2** *For all $\overline{P} \in \mathcal{P}_N$, $d \in \mathcal{R}_+$ and $a \in Act$, whenever $\overline{x} \doteq \overline{y}$, then*

1. $\overline{P}^{\overline{x}} \xrightarrow{a} \overline{P'}^{\overline{x'}}$ *for some $\overline{P'}$ and $\overline{x'}$, implies* $\overline{P}^{\overline{y}} \xrightarrow{a} \overline{P'}^{\overline{y'}}$ *for some $\overline{y'} \doteq \overline{x'}$ and*
2. $\overline{P}^{\overline{x}} \xrightarrow{\epsilon(d)} \overline{P}^{\overline{x}+d}$ *implies* $\overline{P}^{\overline{y}} \xrightarrow{\epsilon(e)} \overline{P}^{\overline{y}+e}$ *and $\overline{x} + d \doteq \overline{y} + e$ for some $e \in \mathcal{R}_+$.*

**Proof:** It is given in the full version of the paper [LW93]. □

Now, we are ready to state the partition theorem, which asserts that the infinite state-space of integer processes can be divided into equivalence classes according to time regions. In fact, many of such classes belong to a large equivalence class and the number of such classes is finite.

**Theorem 3** *(partition) Whenever $\overline{x} \doteq \overline{y}$, then $\overline{P}^{\overline{x}} \dot{\sim} \overline{P}^{\overline{y}}$ for all $\overline{P} \in \mathcal{P}_N$.*

**Proof:** By lemma 2, it should be obvious that the relation: $\mathcal{S} = \{(\overline{P}^{\overline{x}}, \overline{P}^{\overline{y}}) \mid \overline{x} \doteq \overline{y}, \overline{P} \in \mathcal{P}_N\}$ is a strong time abstracted bisimulation. □

In the next section, we want to find a representative state for each equivalence class and then construct a symbolic transition system in terms of the representative states. In order to do so, we need first find a representative point for each time region of $\mathcal{R}_+^n$ for a given $n$.

Let $\mathcal{N}$ denote the naturals. We define the set of grids with granularity $1/k$: $\mathcal{N}_k = \{m/k \mid m \in \mathcal{N}\}$ ranged over by $g, h$ and the set of grid points with granularity $1/k$: $\mathcal{N}_k^n = \{\overline{r} \mid 1 \leq i \leq n, r_i \in \mathcal{N}_k\}$ ranged over by $\overline{r}, \overline{s}$. An obvious choice is to use the the grid points $\mathcal{N}_m^n$ as representative points for $\mathcal{R}_+^n$, for some fixed granularity $1/m$.

[14] $\overline{P}^{|\overline{x}|} = \{\overline{P}^{\overline{y}} \mid \overline{y} \doteq \overline{x}\}$.

We claim that the grid points with granularity $1/(n+1)$ are enough to represent the $n$–dimensional time points $\mathcal{R}_+^n$, that is:

**Lemma 3** *For all $\overline{x} \in \mathcal{R}_+^n$, there exists $\overline{r} \in \mathcal{N}_{n+1}^n$ such that $\overline{x} \doteq \overline{r}$.*

**Proof:** It is given in the full version of the paper [LW93]. □

Clearly, the lemma above will hold for any granularity finer than $1/(n+1)$ such as $1/(n+2), 1/(n+3)$ etc. However, it doesn't hold for a granularity coarser than $1/(n+1)$. To see this, consider the case of $n = 2$: with the granularity $1/2$ one can not find a grid point representing $(1/3, 2/3)$.

Thus $1/(n+1)$ is the coarsest granularity allowing any time region in the $n$–dimensional time space to be represented up to $\doteq$. However, we need a slightly finer granularity (which is in fact $1/(n+2)$ as shown in the following lemma) in order for a region to reach all regions by grid–valued delays, which are reachable by real–valued delays. The following lemma will be heavily used in proving the decidability results.

**Lemma 4** *For all $\overline{r} \in \mathcal{N}_{n+2}^n$ and all $d \in \mathcal{R}_+$, there exist $\overline{r'} \in \mathcal{N}_{n+2}^n$ and $g \in \mathcal{N}_{n+2}$ such that $\overline{r} \doteq \overline{r'}$ and $\overline{r} + d \doteq \overline{r'} + g$.*

**Proof:** It is given in the full version of the paper [LW93]. □

Note that $\overline{r'} + g \in \mathcal{N}_{n+2}^n$, which will prove an essential property for the applicability of our finitary, symbolic semantics to follow. Also, note that it is not always possible to choose $\overline{r'} = \overline{r}$. To see this, consider the case of $n = 2$, $\overline{r} = (3/4, 0)$ and $d = 1/8$. The only possible choices for $g$ is 0 and $1/4$. However in both cases we see that $\overline{r} + g \not\doteq \overline{r} + d$. However, taking $\overline{r'} = (1/2, 0)$ and $g = 1/4$ we obtain as desired $\overline{r'} \doteq \overline{r}$ and $\overline{r'} + g \doteq \overline{r} + d$.

$$\frac{}{a.P \xrightarrow{a}_k P} \qquad \frac{P \xrightarrow{a}_k P'}{\epsilon(0).P \xrightarrow{a}_k P'}$$

$$\frac{P \xrightarrow{a}_k P'}{P+Q \xrightarrow{a}_k P'} \qquad \frac{Q \xrightarrow{a}_k Q'}{P+Q \xrightarrow{a}_k Q'} \qquad \frac{P \xrightarrow{a}_k P'}{X \xrightarrow{a}_k P'}\ [X \overset{def}{=} P]$$

$$\frac{P \xrightarrow{a}_k P'}{P|Q \xrightarrow{a}_k P'|Q} \qquad \frac{Q \xrightarrow{a}_k Q'}{P|Q \xrightarrow{a}_k P|Q'} \qquad \frac{P \xrightarrow{\alpha}_k P' \quad Q \xrightarrow{\bar{\alpha}}_k Q'}{P|Q \xrightarrow{\tau}_k P'|Q'}$$

Table 4: Action Rules for $k$–Semantics.

$$\frac{}{\mathrm{nil} \xrightarrow{\chi}_k \mathrm{nil}} \qquad \frac{}{\epsilon(r + \frac{1}{k}).P \xrightarrow{\chi}_k \epsilon(r).P} \qquad \frac{P \xrightarrow{\chi}_k P'}{\epsilon(0).P \xrightarrow{\chi}_k P'}$$

$$\frac{}{\alpha.P \xrightarrow{\chi}_k \alpha.P} \qquad \frac{P \xrightarrow{\chi}_k P' \quad Q \xrightarrow{\chi}_k Q'}{P+Q \xrightarrow{\chi}_k P'+Q'} \qquad \frac{P \xrightarrow{\chi}_k P'}{X \xrightarrow{\chi}_k P'}\ [X \overset{def}{=} P]$$

$$\frac{P \xrightarrow{\chi}_k P' \quad Q \xrightarrow{\chi}_k Q'}{P|Q \xrightarrow{\chi}_k P'|Q'}\ [P|Q \not\xrightarrow{\tau}_k]$$

Table 5: Delay Rules for $k$–Semantics.

## 5.2 Time–Step Semantics: Sampling

The timed semantics describes how a process will behave at every real-valued time point with arbitrarily fine precision. This introduces the infinite–stateness of timed processes.

In practice, the "sampling" technique is often used to analyze a system. Instead of doing experiment on the system under consideration at *every* time point, only certain *typical* time points are chosen to capture or approximate the full system behaviour. Based on this idea, we develop a time–step semantics called $k$–semantics relativized by the granularity $1/k$, which describes how a process shall behave in every $1/k$ units of time. To achieve finer precision, we can choose a finer granularity. However, the timed processes will be finite–state for any fixed granularity $1/k$. As we shall see latter it is possible to completely capture time abstracted equivalences by sampling with a sufficiently fine granularity. In fact, the granularity required turns out to be $1/(n+2)$ where $n$ is the number of parallel components.

We present the inference rules for the $k$–semantics in two steps: rules for real actions in table 4 and rules for delays in table 5. Note that apart from the index $k$ associated with the arrow, the action rules are the same as in table 1 and the delay rules are parameterized with $k$.

We claim that the processes $\mathcal{P}_N$ are finite-state w.r.t. the transition relation $\longrightarrow_k$ for any non-zero natural $k$. This can be established based on the following facts on processes:

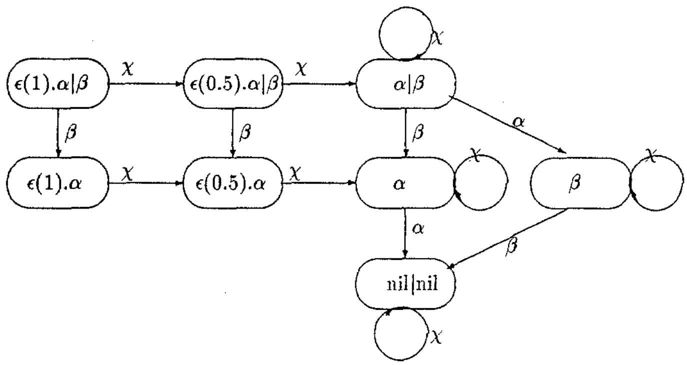


Figure 3: Transition Graph for $\epsilon(1).\alpha.\text{nil}|\beta.\text{nil}$ with Granularity 1/2.

- There is no infinite summation allowed;
- All recursive definitions are well-guarded;
- No parallel composition occurs within a recursion;
- Every process $\overline{P}$ must be *time–stable* after some maximal delay $d_m$, in the sense that $\overline{P}^{d_m} \xrightarrow{\epsilon(d)} \overline{P}^{d_m}$ [15] for all $d$ or $\overline{P}^{d_m} \xrightarrow{\tau}$.

**Example.** In figure 3, we have a transition graph for $\epsilon(1).\alpha.\text{nil}|\beta.\text{nil}$ with granularity 1/2. For clarity, we have omitted nil in the graph. □

[15]Here, $\overline{P}^{d_m}$ stands for $P_1^{d_m}|\ldots|P_n^{d_m}$.

## 5.3 Symbolic Time Abstracted Bisimulation

We shall use a grid state $\overline{P}^{\overline{r}}$ to stand for an equivalence class of (real-valued) states. More precisely, we define:

$$\overline{P}^{|\overline{r}|} = \{\overline{P}^{\overline{x}} \mid \overline{x} \doteq \overline{r}\}$$

A class like $\overline{P}^{|\overline{r}|}$ shall be called as a *symbolic state* (or a symbolic process). We shall use $R, S$ to denote symbolic states. Now, we define a symbolic transition relation $\longmapsto_k$ over symbolic states —(*classes* of real-valued states) as follows:

**Definition 11** *For $\overline{r}, \overline{s} \in \mathcal{N}_k^n$ and $\overline{P}, \overline{Q} \in \mathcal{P}_N$,*

1. $\overline{P}^{|\overline{r}|} \xmapsto{\chi}_k \overline{P}^{|\overline{s}|}$ *if* $\overline{P}^{\overline{r}} \xrightarrow{\chi}_k \overline{P}^{\overline{s}}$
2. $\overline{P}^{|\overline{r}|} \xmapsto{a}_k \overline{Q}^{|\overline{s}|}$ *if* $\overline{P}^{\overline{r}} \xrightarrow{a}_k \overline{Q}^{\overline{s}}$ □

Intuitively, if there is a real transition in the $k$–semantics between two grid states, then there is a symbolic transition between the two equivalence classes they represent. Note that the definition above contains much more information than it looks. In fact, according to the definition, we can infer a symbolic transition like $\overline{P}^{|\overline{r}|} \xmapsto{\chi}_k \overline{P}^{|\overline{s}|}$ whenever $\overline{P}^{\overline{r}'} \xmapsto{\chi}_k \overline{P}^{\overline{s}'}$ for some grid states $\overline{r}' \doteq \overline{r}$ and $\overline{s}' \doteq \overline{s}$. However, the numbers of grid states in $\overline{P}^{|\overline{r}|}$ and $\overline{P}^{|\overline{s}|}$ are finite and hence, the symbolic processes are finite–state w.r.t the symbolic transition relation $\longmapsto_k$.

Like in defining time abstracted equivalence, we now abstract away from the symbolic time steps between symbolic states.

**Definition 12**

1. $R \overset{\varepsilon}{\longrightarrow\!\!\circ}_k S$ *if* $R(\xmapsto{\chi}_k)^* S$
2. $R \overset{a}{\longrightarrow\!\!\circ}_k S$ *if* $R \overset{\varepsilon}{\longrightarrow\!\!\circ}_k \xmapsto{a}_k \overset{\varepsilon}{\longrightarrow\!\!\circ}_k S$ □

**Definition 13** *(strong symbolic $k$–equivalence) A binary relation $\mathcal{S}$ over symbolic states is a strong $k$–simulation if $(R, S) \in \mathcal{S}$ implies that for all $a \in \mathcal{A}ct$ and $\chi$,*

1. *Whenever $R \xmapsto{a}_k R'$ then, for some $S'$, $S \overset{a}{\longrightarrow\!\!\circ}_k S'$ and $(R', S') \in \mathcal{S}$*
2. *Whenever $R \xmapsto{\chi}_k R'$ then, for some $S'$, $S \overset{\varepsilon}{\longrightarrow\!\!\circ}_k S'$ and $(R', S') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a strong $k$-bisimulation if it is symmetrical. The largest strong $k$–bisimulation is called strong symbolic $k$–equivalence, denoted $\mathring{\sim}_k$.*
*We define $\overline{P}^{\overline{r}} \mathring{\sim}_k \overline{Q}^{\overline{s}}$ whenever $\overline{P}^{|\overline{r}|} \mathring{\sim}_k \overline{Q}^{|\overline{s}|}$.* □

Note that $\mathring{\sim}_k$ is decidable for any fixed $k$ because of the finite–stateness of symbolic processes. The following is the main result of this section.

**Theorem 4** *For all $\overline{P}, \overline{Q} \in \mathcal{P}_N$ and $\overline{r}, \overline{s} \in \mathcal{N}_{n+2}^n$, $\overline{P}^{\overline{r}} \dot{\sim} \overline{Q}^{\overline{s}}$ if and only if $\overline{P}^{\overline{r}} \mathring{\sim}_{n+2} \overline{Q}^{\overline{s}}$, where $n$ is the maximal number of components of $\overline{P}$ and $\overline{Q}$* [16].

**Proof:** For the direction: *Only If*, we show that the relation: $\mathcal{R} = \{(\overline{P}^{|\overline{r}|}, \overline{Q}^{|\overline{s}|}) \mid \overline{r}, \overline{s} \in \mathcal{N}_{n+2}^n,\ \overline{P}, \overline{Q} \in \mathcal{P}_N$ and $\overline{P}^{\overline{r}} \dot{\sim} \overline{Q}^{\overline{s}} \mid \}$ is a strong symbolic $(n+2)$–bisimulation; for the other direction, we show

[16] Note that we can always extend $\overline{P}$ or $\overline{Q}$ with nil–processes as auxiliary components so that they own the same number of components.

that the relation: $\mathcal{S} = \{(\overline{P}^{\overline{r}}, \overline{Q}^{\overline{s}}) \mid \overline{r}, \overline{s} \in \mathcal{N}^n_{n+2},\ \overline{P}, \overline{Q} \in \mathcal{P}_N$ and $\overline{P}^{|\overline{r}|} \overset{\circ}{\sim}_{n+2} \overline{Q}^{|\overline{s}|} \mid \}$ is a strong time abstracted bisimulation up to $\overset{\bullet}{\sim}$. A complete proof is given in the full version of the paper [LW93]. □

We extend the results to weak time abstracted equivalence.

**Definition 14**

1. $R \overset{\varepsilon}{\Longrightarrow}\!\!\circ_k S$ *if* $R(\overset{\chi}{\longmapsto}_k \cup \overset{\tau}{\longmapsto}_k)^* S$
2. $R \overset{\alpha}{\Longrightarrow}\!\!\circ_k S$ *if* $R \overset{\varepsilon}{\Longrightarrow}\!\!\circ_k \overset{\alpha}{\longmapsto}_k \overset{\varepsilon}{\Longrightarrow}\!\!\circ_k S$ □

**Definition 15** *(weak symbolic k–equivalence) A binary relation $\mathcal{S}$ over symbolic states is a weak k–simulation if $(R, S) \in \mathcal{S}$ implies that for all $\alpha \in \mathcal{A}ct - \{\tau\}$ and $\theta \in \{\chi, \tau\}$,*

1. *Whenever $R \overset{\alpha}{\longmapsto}_k R'$ then, for some $S'$, $S \overset{\alpha}{\Longrightarrow}\!\!\circ_k S'$ and $(R', S') \in \mathcal{S}$*
2. *Whenever $R \overset{\theta}{\longmapsto}_k R'$ then, for some $S'$, $S \overset{\varepsilon}{\Longrightarrow}\!\!\circ_k S'$ and $(R', S') \in \mathcal{S}$*

*We call such a simulation $\mathcal{S}$ a weak k–bisimulation if it is symmetrical. The largest weak k–bisimulation is called weak symbolic k–equivalence, denoted $\overset{\circ}{\approx}_k$.*
*We define $\overline{P}^{\overline{r}} \overset{\circ}{\approx}_k \overline{Q}^{\overline{s}}$ whenever $\overline{P}^{|\overline{r}|} \overset{\circ}{\approx}_k \overline{Q}^{|\overline{s}|}$.* □

Finally, we achieve the decidability result for weak time abstracted equivalence.

**Theorem 5** *For all $\overline{P}, \overline{Q} \in \mathcal{P}_N$ and $\overline{r}, \overline{s} \in \mathcal{N}^n_{n+2}$, $\overline{P}^{\overline{r}} \overset{\bullet}{\approx} \overline{Q}^{\overline{s}}$ if and only if $\overline{P}^{\overline{r}} \overset{\circ}{\approx}_{n+2} \overline{Q}^{\overline{s}}$, where $n$ is the maximal number of components of $\overline{P}$ and $\overline{Q}$.*

**Proof:** It is similar to the proof for theorem 4. A complete proof is given in the full version of the paper [LW93]. □

# 6 Conclusion

In this paper we have introduced a notion of *time–abstracting* bisimulation equivalence.

As the first main result of this paper, we have demonstrated that two processes are interchangeable in any context up to time–abstracted equivalence precisely when they are timed equivalent. Thus, by resorting to *implicit* specifications — i.e. specifications of a system in contexts — we may reveal *all* timing properties of a system.

As our second main result we have established the decidab of the time–abstracted equivalence by providing a finite–state and symbolic yet structured, operational semantics of processes. The symbolic semantics can be seen as sampling a process with a given frequency; we prove that sufficiently frequent sampling — $1/(n+2)$ where $n$ is the number of parallel components — yields a symbolic equivalence completely capturing the time–abstracted equivalence.

The minimization algorithm presented in [ACH92] can be seen to minimize timed graphs [AD90] with respect to time–abstract bisimulation equivalence even though no notion of time–abstracted bisimulation is given in the paper. Despite the purpose of the minimization effort being to obtain more efficient model–checking algorithms with respect to a real–time temporal logic, we believe that the results of [ACH92] can provide an alternative method for deciding time–abstracted equivalences. However, we are of the opinion that our approach is simpler (certainly from a process algebraic point of view) as it is based directly on a traditional structured, operational semantics.

Recently, we have completed a prototype implementation of a tool–set for timed and time–abstracted bisimulation equivalences based on the methods described in this paper and in [Č92]. In addition the tool–set applies the efficient, local checking technique described in [La92], thus avoiding to explore the state–space more than necessary. We hope to report upon this work in a forthcoming paper [CGL92].

# References

[ACD90] Rajeev Alur, Costas Courcoubetis, and David Dill. Model–checking for real–time systems. In Proceedings of the Fifth IEEE Symposium on Logic in Computer Science, 1990.

[ACH92] R. Alur, C Courcoubetis, N. Halbwachs, D. Dill, H. Wong–Toi. Minimization of Timed Transition Systems. CONCUR92, LNCS 630, 1992.

[AD90] Rajeev Alur and David Dill. Automata for modelling real–time systems. In Automata, Languages and Programming: Proceedings of the 17th ICALP, LNCS 443. Springer-Verlag, 1990.

[BB89] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. Technical Report P8916, University of Amsterdam, 1989.

[CGL92] K. Cerans, J.C. Godskesen, K.G. Larsen. JANUS: a tool for analyzing real–time processes, Aalborg University, (in preparation), 1992.

[Che91b] Liang Chen. An interleaving model for real–time systems. Technical report, LFCS, University of Edinburgh, Scotland, 1991. Preliminary version.

[CPS89] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench. Technical report, LFCS, University of Edinburgh, Scotland, 1989.

[DS89] Jim Davis and Steve Schneider. An introduction to timed CSP. Technical Report PRG–75, Oxford University Computing Laboratory, 1989.

[FK91] W.J. Fokkink and S. Klusener. Real time algebra with prefixed integration. Technical report, CWI, Amsterdam, 1991.

[GL92] Jens Chr. Godskesen and Kim G. Larsen. Real–time calculi and expansion theorems. In Twelfth Conference on the FST and TCS, Lecture Notes in Computer Science. Springer-Verlag, December 1992. To appear.

[HLW91] Uno Holmer, Kim Larsen, and Yi Wang. Deciding properties of regular timed processes. In the proceedings of CAV91, volume 575 of Lecture Notes in Computer Science. Springer-Verlag, 1991.

[HNJ92] T. Henzinger, X. Nicollin, J. Sifakis, and J. Voiron. Symbolic Model Checking for Real–Time Systems. Proceedings of the 7th IEEE Symposium on Logic in Computer Science, 1992.

[Hoa85] C.A.R. Hoare. Communicating Sequential Processes. Prentice–Hall, 1985.

[HR91] Matthew Hennessy and Tim Regan. A process algebra for timed systems. Technical Report 5/91, University of Sussex, 1991.

[Jef91b] Allan Jeffrey. A linear time process algebra. In the proceedings of CAV91, volume 575 of Lecture Notes in Computer Science. Springer-Verlag, July 1991.

[JGZ89] K.G. Larsen J.C. Godskesen and M. Zeeberg. Tav — tools for automatic verification — users manual. Technical Report R 89-19, Department of Mathematics and Computer Science, Aalborg University, 1989. Presented at workshop on Automatic Methods for Finite State Systems, Grenoble, France, Juni 1989.

[KS90] P.C. Kanellakis and S.A. Smolka, CCS Expressions, finite state processes, and three problems of equivalence. Information and Control Vol 86, 1990.

[La92] Kim G. Larsen. Efficient Local Correctness Checking In Proceedings of CAV92, Montreal, Canada.

[LW93] Kim G. Larsen and Wang Yi. Time Abstracted Bisimulation: Implicit Specifications and Decidability. Tech Report, Department of Computer Systems, Uppsala University, Sweden, 1993.

[Mil89] Robin Milner. Communication and Concurrency. Series in Computer Science. Prentice-Hall International, 1989.

[MT90] Faron Moller and Chris Tofts. A temporal calculus of communicating systems. In CONCUR'90, volume 458 of Lecture Notes in Computer Science. Springer-Verlag, 1990.

[NRSV90] Thomas A. Henzinger, X. Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic Model Checking for Real-Time Systems IEEE Proc. 7th Sym. Logic in Computer Science, California, June, 1992.

[NRSV90] X. Nicollin, J.-L. Richier, Joseph Sifakis, and J. Voiron. ATP: an algebra for timed processes. In Proceedings of the IFIP TC 2 Working Conference on Programming Concepts and Methods, Sea of Gallilee, Israel, April 1990.

[NSY91] Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. From ATP to timed graphs and hybrid systems. In Real-Time: Theory in Practice, volume 600 of Lecture Notes in Computer Science. Springer-Verlag, 1991.

[RH92] Rajeev Alur and T. Henzinger. Logics and Models of Real Time: a Survey. REX Workshop on Real Time: Theory and Practice, 1991.

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

[PT87] Paige and Tarjan. Three partition refinement algorithms. SIAM Journal of Computing, 16(6), 1987.

[Sch91] Steve Schneider. An operational semantics for timed CSP. April 1991.

[Č92] Kārlis Čerāns. Decidability of bisimulation equivalences for processes with parallel timers. To appear in Proceedings of CAV'92, 1992.

[SV89] R. De Simone and D. Vergamini. Aboard AUTO. Technical Report 111, INRIA, Sofia-Antipolis, 1989.

[Wan90] Yi Wang. Real-time behaviour of asynchronous agents. In CONCUR '90, volume 458 of Lecture Notes in Computer Science. Springer-Verlag, 1990.

[Wan91a] Yi Wang. A Calculus of Real Time Systems. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 1991.

[Wan91b] Yi Wang. CCS + time = an interleaving model for real time systems. In ICALP91, LNCS 510. Springer-Verlag, 1991.