Time-Action Alternating Model for Timed LOTOS and its Symbolic Verification of Bisimulation Equivalence

Akio Nakata, Teruo Higashino, and Kenichi Taniguchi
Department of Information and Computer Sciences, Osaka University,
Machikaneyama 1-3, Toyonaka, Osaka 560, Japan.
Phone:+81-6-850-6608. Fax:+81-6-850-6609.
E-mail:{nakata,higashino,taniguchi}@ics.es.osaka-u.ac.jp

Abstract
Verification of timed bisimulation equivalence is generally difficult because of state explosion caused by concrete time values. In this paper, we propose a verification method to verify timed bisimulation equivalence of two timed processes using a symbolic technique similar to (Hennessy and Lin 1995). We first propose a new model of timed processes, Alternating Timed Symbolic Labelled Transition System(A-TSLTS). In A-TSLTS, each state has some parameter variables and those values determine its behaviour. Each transition in an A-TSLTS has a guard predicate. The transition is executable if and only if its guard predicate is true under specified parameter values. In the proposed method, we can obtain the weakest condition for a state-pair in a finite A-TSLTS to make the state-pair be timed bisimulation equivalent. We also show that this result can be applied to the language LOTOS/T(Nakata et al. 1994), a timed extension of LOTOS(ISO 1989).

Keywords
Real-time and probability aspects of FDTs; Verification, validation and testing; LOTOS; A-TSLTS; Symbolic bisimulation

1 INTRODUCTION

Verification of timed bisimulation equivalence for timed processes is generally difficult because of state explosion. There are some proposals to solve this problem(Holmer et al. 1991; Čerãns 1992; Chen 1992; Alur et al. 1994). But they all have some restrictions in describing time constraints of actions. On the other hand, for data-passing processes, a verification method of bisimulation equivalence is proposed(Hennessy and Lin 1995). This method has some advantages: (1). Its verification cost does not depend on data domain nor absolute values of constants which we use in data constraints, and (2). the method does not depend on predicates which we choose for describing data constraints (although they should be decidable in order to verify the equivalence). It is desirable that there exists a method to verify timed bisimulation equivalence which has the same advantages as above.

In this paper, first, we propose a new model for describing timed processes, and then we propose a method to verify timed bisimulation equivalence of two timed processes in the proposed model using the technique so-called 'symbolic bisimulation'(Hennessy and Lin 1995).

A new model, Alternating Timed Symbolic Labelled Transition System(A-TSLTS, for short), is introduced to describe time-constrained processes. Each state in an A-TSLTS may have some parameter variables(eg. x, y). Each transition in an A-TSLTS has a guard predicate such as 'execute the transition a between $x + 5$ and y seconds from now.' The guard predicate of a transition may contain any parameter variables associated to its source state, any numerical operations on time domain, and any atomic predicates. We can use any logic. The logic only needs to be decidable in order to verify the equivalence in the proposed method. In this paper, only timed transitions are considered (data-passing is ignored).

We model a time transition by a delay transition $\xrightarrow{e(d)}$ with a delay variable d, which stands for an amount of the delay (duration). This is the same as (Holmer et al. 1991). This modeling has an advantage that we can treat durations equally as input/output data. So, although we only handle time here, we can easily extend the result to the model which handles both time and data-passing. Each delay transition $\xrightarrow{e(d)}$ and action transition $\xrightarrow{a}$ have guard predicates which may contain the delay variables and the parameter variables at their source states (they possibly include some delay variables in former delay transitions). We refer to such a model as "Timed Symbolic Labelled Transition System(TSLTS)."

It is difficult to consider symbolic bisimulation(Hennessy and Lin 1995) on a TSLTS. The reason is as follows. A delay transition $\xrightarrow{e(d)}$ whose amount of delay is d, is equivalent to a sequence of delay transitions $\xrightarrow{e(d_1)}\xrightarrow{e(d_2)}\cdots\xrightarrow{e(d_n)}$ where $d_1 + d_2 + \cdots + d_n = d$. Also, in a TSLTS, it is possible that after $\xrightarrow{e(d_1)}$ is executed, both $\xrightarrow{e(d_2)}$ and $\xrightarrow{a}$ may be executable. So, in general, the sequence $\xrightarrow{e(d_1)}\xrightarrow{e(d_2)}$ is not easily reduced to one transition. In order to make a matching between two transitions which form a bisimulation, we must make a (possibly infinitely many) sequence-to-sequence matching, which makes the problem difficult. Therefore, in this paper, we assume our model to have *alternating* property. Each state of a TSLTS must belong to one of the two kinds of sets of states, the one is a set of *idle states*, and the other is a set of *active states*. From an idle state, only a delay transition is possible and then it moves to an active state. From an active state, some action transitions are possible. After one of them is executed, it comes back to an idle state. We call such a restricted TSLTS as an Alternating TSLTS (A-TSLTS). In the A-TSLTS model, we can make the bisimulation matching of delay transitions to one-to-one. Consecutive execution of actions (without delay) can be expressed in an A-TSLTS by inserting a delay transition of zero duration between two action transitions.

Using a similar algorithm as (Hennessy and Lin 1995), from a given state-pair we obtain the weakest condition (similar to (Hennessy and Lin 1995), we refer to the condition as *most general boolean*, *mgb* for short) to make the two states be timed bisimulation equivalent. For example, let us consider the following two processes, P and Q. The process

P may execute the action a between $x+5$ and y seconds from now, or execute the action b between y and $x+10$ seconds from now. The process Q may execute the action a between 10 and z seconds from now. In order to make P and Q bisimilar, the condition "$(x+5=10) \wedge (y=z) \wedge (y > x+10)$" must hold (if $(y > x+10)$, then P cannot execute the action b). On the other hand, the condition is also a sufficient condition to make P and Q bisimilar. Such a condition is the mgb. In the proposed method, even if P and Q are infinite processes, if the corresponding A-TSLTS has finite states and finite variables, we can obtain the mgb for any two states. Once we obtain the mgb, we can verify whether the two states are timed bisimulation equivalent w.r.t. specified parameter values by checking whether the values satisfy the mgb.

The proposed algorithm takes an A-TSLTS and its state-pair as an input, and it outputs the mgb for the state-pair. We also show that the algorithm can easily be extended to verify *untimed bisimulation equivalence*(Nakata et al. 1994), which is a bisimulation equivalence where we allow the executed time of actions does not have to be precisely equal. The notion of untimed bisimulation equivalence is essentially identical to *time abstracted bisimulation* in (Larsen and Wang 1993; Alur et al. 1994).

The method can also apply to structural process description languages such as CCS or LOTOS. In this case, we have only to provide a transformation from an expression of the language to an A-TSLTS. In this paper, we will apply the method to LOTOS/T, a timed extension of LOTOS defined in (Nakata et al. 1994).

This paper is organized as follows. In Section 2, the model of timed processes, A-TSLTS, is defined. In Section 3, timed bisimulation equivalence of states in an A-TSLTS is defined. In Section 4, an algorithm is presented to construct the mgb for two states in an A-TSLTS w.r.t. timed bisimulation equivalence. In Section 5, untimed bisimulation equivalence of states in an A-TSLTS is defined and an extension of the algorithm to verify untimed bisimulation equivalence is presented. In Section 6, we apply our verification method to the verification of LOTOS/T expressions. Finally, in Section 7, we conclude this paper.

2 TSLTS MODEL

A *TSLTS* is an LTS where each state s has a set of parameter variables $DVar(s)$, and each transition is either an action transition, represented as $s \xrightarrow{a,P} s'$ or a delay transition represented as $s \xrightarrow{e(d),P} s'$. a is an action name. d is a variable which stands for a duration. Each P is a transition condition. The transition condition P is a formula of a (decidable) 1st-order arithmetics on any (dense or discrete) time domain. P may contain any variables in $DVar(s)$ (s is a source state of the transition). In a delay transition $s \xrightarrow{e(d),P} s'$, P may also contain the variable d.

Intuitively, a delay transition $s \xrightarrow{e(d),P} s'$ represents a state-transition only by delay. Its duration is d and d must satisfy P under a current assignment for other parameter variables in $DVar(s)$. The delay is possible up to the maximum value of d's which satisfy P. The delay over the maximum value of d is not allowed (time-deadlock(Moller and Tofts 1990),urgency(Bolognesi and Lucidi 1992)). When the transition is completed, the actual duration (which satisfies P) is assigned to the variable d. $DVar(s')$ may contain the variable d. So the value of d may be used in conditions of any succeeding transitions. An action transition $s \xrightarrow{a,P} s'$ represents an execution of an action a when P holds under a current

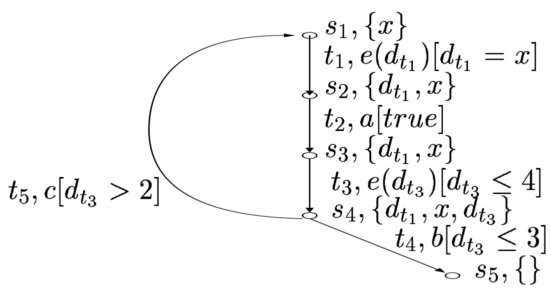

Figure 1 Example of TSLTS

assignment for parameter variables in $DVar(s)$. The execution of an action is considered instantaneous, since we take interleaving semantics to express concurrency(Wang 1991; Chen 1992). The state s may have multiple outgoing action transitions. In that case, one of executable action transitions is nondeterministically chosen and then executed.

Example 1 We show an example of a TSLTS in Fig. 1. In Fig. 1, for convenience, the names $s_1, s_2, \ldots$ are assigned for states and $t_1, t_s, \ldots$ for transitions. The set associated to each state s_i represents $DVar(s_i)$. $a[P]$ (or $e(d)[P]$) associated to each transition represents an action name a (or a delay with its duration of d, respectively) with a transition condition P. When a value v is assigned to the parameter variable x at state s_1, the TSLTS in Fig. 1 behaves as follows. First, $x = v$ units of time is elapsed (the value v is assigned to d_{t_1}) and then the action a is executed. Next, before 4 units of time elapse, the action b or c is executed. The action b is executable when the duration is within 3 units of time. The action c is also executable when the duration is more than or equal to 2 units of time. In the case that c is executed, the TSLTS moves its state to s_1 and then repeats the behaviour from the beginning. □

In order to make it easier to consider symbolic bisimulation, we restrict a TSLTS so that its states fall into two categories of states, idle states and active states. Each idle state has only a delay transition as an outgoing transition and the destination is an active state. An active state has only action transitions as outgoing transitions and all the destinations are idle states. We call this restricted TSLTS as an *Alternating TSLTS (A-TSLTS)*. The notion of A-TSLTS is inspired by (Hansson 1991).

In the rest of this paper, we assume that each TSLTS is an A-TSLTS, and it is time-deterministic, i.e., every state has at most one outgoing delay transition. Time-determinacy is a reasonable assumption when we consider processes of real-world. Many other studies also assume time-determinacy(Moller and Tofts 1990; Wang 1991; Chen 1992).

Example 2 The TSLTS of Example 1 is an A-TSLTS because a division into $\{s_1, s_3, s_5\}$ (idle states) and $\{s_2, s_4\}$ (active states) is possible. It is also time-deterministic.

3 TIMED BISIMULATION EQUIVALENCE

In this section, we define timed bisimulation equivalence for A-TSLTS. Before all, we need some preliminary definitions.

Definition 1 ● We denote *assignments* of values to variables by $\rho, \rho', \ldots$.

● For a predicate P and an assignment ρ, we denote $\rho \models P$ iff P is true under an assignment ρ.

- We denote $\rho[x = e]$ is the same assignment as ρ except that the value of the expression e is assigned to the variable x.
- We denote a tuple (s, ρ) of a state s in a TSLTS and an assignment ρ, as $\rho(s)$. $\rho(s)$ stands for a state with some parameter *values*(not variables) associated to s. We call it an *instance* of s w.r.t. ρ. $\square$

The actual moves of a TSLTS are formally defined by considering the corresponding (traditional) LTS, whose states are all instances of TSLTS states, and whose transitions are labelled by either an action name or a concrete value of a duration.

Definition 2 For a TSLTS M, its corresponding *semantic LTS* M' is defined as follows:

- The set of states in M' are the set of all instances of M, i.e. $\{\rho(s) | \rho\text{:an assignment}, s\text{:a state of } M\}$.
- Each transition in M' is labelled by either an action name a of M, or any non-negative time value t.
- For each transition $s \xrightarrow{a,P} s'$ in M and each assignment ρ, M' has a transition $\rho(s) \xrightarrow{a} \rho(s')$ iff $\rho \models P$.
- For each transition $s \xrightarrow{e(d),P} s'$ in M, each assignment ρ, and any non-negative time value t, M' has a transition $\rho(s) \xrightarrow{t} \rho[d = t](s')$ iff $\rho[d = t] \models \exists d'[d \leq d' \wedge P\{d'/d\}]$ ($P\{d'/d\}$ denotes P whose any occurrence of a free variable d is replaced by d'). Moreover, for any non-negative time value t' which satisfies $t' \leq t$, M' has a transition $\rho[d = t'](s') \xrightarrow{t-t'} \rho[d = t](s')$. $\square$

Remark: The predicate "$\exists d'[d \leq d' \wedge P\{d'/d\}]$" means that P holds at some duration d' where $d \leq d'$. In such a case, a delay of the duration d (as well as d') is allowed.

The method for modeling real-time processes by considering a delay transition with an associated time value is similar to (Wang 1991; Holmer et al. 1991; Chen 1992).

For a given TSLTS, timed bisimulation equivalence of its two instances of states is defined by considering a traditional bisimulation equivalence on its semantic LTS.

Definition 3 A *timed bisimulation relation* R is a binary relation on a set of instances of TSLTS states $\{\rho(s) | s\text{:a TSLTS state}, \rho\text{:an assignment}\}$, which satisfies the following conditions:

- R is a symmetric relation, and
- if $(\rho_i(s_i), \rho_j(s_j)) \in R$, then all of the following conditions hold:
  - For any time value t, if $\rho_i(s_i) \xrightarrow{t} \rho'_i(s'_i)$, then there exist some s'_j and ρ'_j such that $\rho_j(s_j) \xrightarrow{t} \rho'_j(s'_j)$ and $(\rho'_i(s'_i), \rho'_j(s'_j)) \in R$,
  - For any action name a in the TSLTS, if $\rho_i(s_i) \xrightarrow{a} \rho'_i(s'_i)$, then there exist some s'_j and ρ'_j such that $\rho_j(s_j) \xrightarrow{a} \rho'_j(s'_j)$ and $(\rho'_i(s'_i), \rho'_j(s'_j)) \in R$.

If there exists some timed bisimulation equivalence R such that $(\rho_i(s_i), \rho_j(s_j)) \in R$, the two instances $\rho_i(s_i)$ and $\rho_j(s_j)$ are called *timed bisimulation equivalent*, which is denoted by $\rho_i(s_i) \sim_t \rho_j(s_j)$. Especially, if $\rho(s_i) \sim_t \rho(s_j)$, then the two states s_i and s_j are called *timed bisimulation equivalent w.r.t. an assignment* ρ. $\square$

```
$mgb(s_i, s_j) \overset{\text{def}}{=} mgb1(s_i, s_j, \emptyset)$
$mgb1(s_i, s_j, W) \overset{\text{def}}{=}$ if $(s_i, s_j) \in W$ then return true
                else if $(s_i, s_j)$ is a pair of idle states, then return $match\_delay(s_i, s_j, W)$
                else if $(s_i, s_j)$ is a pair of active states, then return $match\_action(s_i, s_j, W)$
                else return false
$match\_delay\ (s_i, s_j, W) \overset{\text{def}}{=}$ if $s_i \xrightarrow{e(d_i), P_i} s_{i'}$ and $s_j \xrightarrow{e(d_j), P_j} s_{j'}$
    then let $\{d = new(DVar(s_i) \cup DVar(s_j)),$
            $M_{i',j'} = mgb1(s_{i'}[d_i \to d], s_{j'}[d_j \to d], W \cup \{(s_i, s_j)\})\}$ in
        return $\forall d[P_i\{d/d_i\} \Rightarrow [P_j\{d/d_j\} \wedge M_{i',j'}]] \wedge \forall d[P_j\{d/d_j\} \Rightarrow [P_i\{d/d_i\} \wedge M_{i',j'}]]$
    else if $s_i \overset{e(d_i), P_i}{\not\longrightarrow}$ and $s_j \overset{e(d_j), P_j}{\not\longrightarrow}$ then return true else return false
$match\_action(s_i, s_j, W) \overset{\text{def}}{=}$ return $\bigwedge_{a \in Act}\{match\_action1(a, s_i, s_j, W)\}$
$match\_action1(a, s_i, s_j, W) \overset{\text{def}}{=}$ let $\{K = \{k | s_i \xrightarrow{a, P_k} s_{i_k}\}$, $L = \{l | s_j \xrightarrow{a, Q_l} s_{j_l}\}$,
            $M_{k,l} = mgb1(s_{i_k}, s_{j_l}, W \cup \{(s_i, s_j)\})\}$ in
        return $\bigwedge_{k \in K}\{P_k \Rightarrow \bigvee_{l \in L}\{Q_l \wedge M_{k,l}\}\} \wedge \bigwedge_{l \in L}\{Q_l \Rightarrow \bigvee_{k \in K}\{P_k \wedge M_{k,l}\}\}$
where, for a set $V$ of variables, $new(V)$ denotes a function which returns an appropriate new
variable $x$ such that $x \notin V$.
```

Figure 2 Algorithm to compute $mgb(s_i, s_j)$.

4 VERIFICATION OF TIMED BISIMULATION EQUIVALENCE

For any state-pair (s_i, s_j) in an A-TSLTS, we call the weakest condition P such that if $\rho \models P$ then s_i and s_j are timed bisimulation equivalent w.r.t. ρ, as the *mgb* of (s_i, s_j). If we can obtain the mgb P for any state-pair (s_i, s_j), then the verification of timed bisimulation equivalence of $\rho(s_i)$ and $\rho(s_j)$ is reduced to the verification to check whether $\rho \models P$.

To keep track of the correspondences between variables during matching, it is useful to replace some different variables of two states with some common name, standing for their matched common value which equates the two states. In order to do so, we consider the mgb for a pair of *terms* instead of states in A-TSLTS. This is similar to (Hennessy and Lin 1995). A *term* is a tuple of a state and a *substitution.* A *substitution* is a mapping from variables to variables. We denote a term (s, σ) as $s\sigma$, where s is a state of A-TSLTS and σ is a substitution. We also denote a substitution which maps the variable d to d' as $[d \to d']$. If σ is an identity substitution, we abbreviate $s\sigma$ to s and we do not distinguish between the term $s\sigma$ and the state s. Note that if the set of variables is a finite set, then the set of all possible substitutions are finite. A transition between terms is defined as $s\sigma \xrightarrow{e(\sigma(d)), P\sigma} s'\sigma$ ($s\sigma \xrightarrow{a, P\sigma} s'\sigma$) iff $s \xrightarrow{e(d), P} s'$ ($s \xrightarrow{a, P} s'$, respectively) in an A-TSLTS. We denote the mgb of a term-pair (s_i, s_j) as $mgb(s_i, s_j)$. If the A-TSLTS has only finite states and finite variables, $mgb(s_i, s_j)$ is obtained by the algorithm in Fig. 2.

The function $mgb(s_i, s_j)$ takes two arguments s_i and s_j, any two states in an A-TSLTS, and returns the mgb for (s_i, s_j). The function $mgb1(s_i, s_j, W)$ takes three arguments s_i, s_j and a set W of state-pairs. W is a set of *already visited* pairs, introduced to make sure the algorithm eventually terminates. For $(s_i, s_j) \in W$, it simply returns *true.* Otherwise, it returns $match\_delay(s_i, s_j, W)$ if (s_i, s_j) is a pair of idle states, or $match\_action(s_i, s_j, W)$ if (s_i, s_j) is a pair of active states. $match\_delay(s_i, s_j, W)$ ($match\_action(s_i, s_j, W)$) is a function which recursively computes the mgb for (s_i, s_j), where we assume (s_i, s_j) is a pair of idle (active, respectively) states.

The function *match_delay*(s_i, s_j, W) computes the mgb for two idle states s_i and s_j as follows. Firstly, from the definition of A-TSLTS and time-determinacy, delay transitions from s_i and s_j correspond to one-to-one, including duration values. So we unifies the delay variables in the two transitions into one. We introduce a new variable d representing the common duration of delay. We choose $d = new(DVar(s_i) \cup DVar(s_j))$. W.r.t. a given assignment ρ, if s_i and s_j are timed bisimulation equivalent, and if any delay transition of duration v from s_i is possible, then there must exist a delay transition of the same duration v from s_j, and the destinations s'_i and s'_j must be timed bisimulation equivalent w.r.t. $\rho[d = v]$. For example, if $s_i \xrightarrow{e(d_i),d_i \le x} s'_i$ and $s_j \xrightarrow{e(d_j),d_j \le y} s_j$, then $\forall d[d \le x \Rightarrow [d \le y \wedge$ (the mgb for $(s'_i[d_i \to d], s'_j[d_j \to d]))]$ holds. Here, in general, the mgb for (s'_i, s'_j) contains the variables d_i or d_j. To preserve the information that d_i and d_j are equal, we consider the mgb for $(s'_i[d_i \to d], s'_j[d_j \to d])$ instead. In general, the mgb for $(s'_i[d_i \to d], s'_j[d_j \to d])$ contains the variable d as a free variable. It represents the mgb for (s'_i, s'_j) in the case $d_i = d_j = d$ is assumed.

The above discussions must also be applied when s_i and s_j are exchanged. Thus, ρ must satisfy the following condition if s_i and s_j are timed bisimulation equivalent w.r.t. ρ.

$$\forall d[P_i\{d/d_i\} \Rightarrow [P_j\{d/d_j\} \wedge M_{i',j'}]] \wedge \forall d[P_j\{d/d_j\} \Rightarrow [P_i\{d/d_i\} \wedge M_{i',j'}]]. \qquad (1)$$

where $M_{i',j'} = mgb(s'_i[d_i \to d], s'_j[d_j \to d])$. On the other hand, if s_i and s_j are not timed bisimulation equivalent w.r.t. ρ, then, for example, a delay transition of some duration v' is possible from s_i, which is impossible on s_j, or otherwise s'_i and s'_j are not equivalent w.r.t. $\rho[d = v'']$ for some value v''. In any case, Expression (1) does not hold. Therefore, Expression (1) is the weakest condition such that ρ must satisfy in order to make $\rho(s_i)$ and $\rho(s_j)$ be timed bisimulation equivalent, i.e., the mgb for (s_i, s_j). The function *match_delay*(s_i, s_j, W) computes $M_{i',j'} = mgb1(s'_i[d_i \to d], s'_j[d_j \to d], \{(s_i, s_j)\})$ recursively (where (s_i, s_j) is treated as a *already visited* pair) and then returns Expression (1) as the mgb for (s_i, s_j).

The function *match_action*(s_i, s_j, W) returns the mgb for active states s_i and s_j, which is computed as follows. Firstly, if s_i and s_j are timed bisimulation equivalent w.r.t. an assignment ρ, for any action a in a set *Act* of all actions, the following condition holds. For any possible transition $s_i \xrightarrow{a,P_k} s_{i_k}$ whose transition condition P_k satisfies $\rho \models P_k$, if the action a is executable, there must exist some transition $s_j \xrightarrow{a,Q_l} s_{j_l}$ whose transition condition Q_l also satisfies $\rho \models Q_l$ and the destinations s_{i_k} and s_{j_l} must be timed bisimulation equivalent w.r.t. ρ (ρ must satisfy the mgb for (s_{i_k}, s_{j_l})). The above discussions must be true when s_i and s_j are exchanged. Therefore, when we let $K = \{k | s_i \xrightarrow{a,P_k} s_{i_k}\}$, $L = \{l | s_j \xrightarrow{a,Q_l} s_{j_l}\}$ and $M_{k,l} = mgb(s_{i_k}, s_{j_l})$, ρ must satisfy

$$\bigwedge_{k \in K} \{P_k \Rightarrow \bigvee_{l \in L} \{Q_l \wedge M_{k,l}\}\} \wedge \bigwedge_{l \in L} \{Q_l \Rightarrow \bigvee_{k \in K} \{P_k \wedge M_{k,l}\}\}. \qquad (2)$$

A conjunction of Expression (2) over all actions $a \in Act$ is a condition such that ρ must satisfy if s_i and s_j are timed bisimulation equivalent for any action w.r.t. ρ. On the other hand, if ρ does not make s_i and s_j be timed bisimulation equivalent, there must exist some action a' such that, for example, $s_i \xrightarrow{a',P_k} s_{i_k}$ is executable and for any l, either $s_j \xrightarrow{a',Q_l} s_{j_l}$ is not executable or s_{i_k} and s_{j_l} are not timed bisimulation equivalent w.r.t. ρ. In any case,

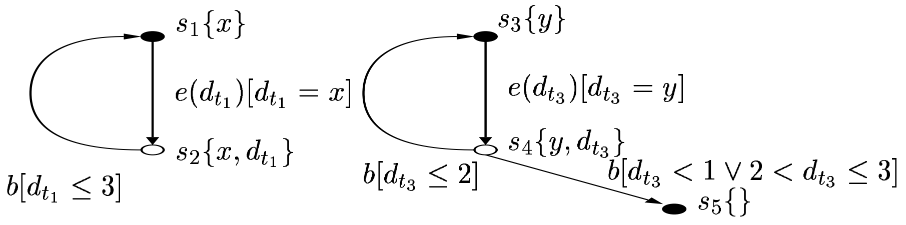

Figure 3 Example of A-TSLTS

Expression (2) does not hold. Therefore, a conjunction of Expression (2) over all actions $a \in Act$ is the weakest condition that ρ must satisfy to make s_i and s_j be timed bisimulation equivalent w.r.t. ρ, i.e. the mgb for (s_i, s_j). The function $match\_action1(a, s_i, s_j, W)$ computes each M_{i_k,j_l} recursively (with (s_i, s_j) as an already visited pair), and then returns Expression (2). The function $match\_action(s_i, s_j, W)$ composes a conjunction of $match\_action1(a, s_i, s_j, W)$ over all $a \in Act$ and returns it as the mgb for (s_i, s_j).

The algorithm $mgb(s_i, s_j)$ terminates if the considered A-TSLTS has only a finite number of states and variables (thus it has only a finite number of pair of terms).

Formally, we obtain the correctness result by the following theorem.

Theorem 1 [Soundness] If $\rho \models mgb(s_i, s_j)$, then $\rho(s_i) \sim_t \rho(s_j)$. □

Theorem 2 [Completeness] If $\rho(s_i) \sim_t \rho(s_j)$, then $\rho \models mgb(s_i, s_j)$. □

The formal proof of the correctness for this algorithm is similar to Appendix B. in (Hennessy and Lin 1995). We omit the detail due to the space restriction.

Example 3 For a pair (s_1, s_3) of the A-TSLTS in Fig. 3, $mgb(s_1, s_3)$ is obtained as follows.

$$mgb(s_1,s_3) = \forall d_1[d_1 = x \Rightarrow [d_1 = y \wedge M_{24}]] \wedge \forall d_1[d_1 = y \Rightarrow [d_1 = x \wedge M_{24}]]$$

where,

$$\begin{aligned} M_{24} &= mgb(s_2[d_{t_1} \to d_1], s_4[d_{t_3} \to d_1], \{(s_1, s_3)\}) \\ &= [d_1 \le 3 \Rightarrow [d_1 \le 2 \wedge M_{13} \vee (d_1 < 1 \vee 2 < d_1 \le 3) \wedge M_{15}]] \wedge \\ &\quad\ [d_1 \le 2 \Rightarrow [d_1 \le 3 \wedge M_{13}]] \wedge [(d_1 < 1 \vee 2 < d_1 \le 3) \Rightarrow [d_1 \le 3 \wedge M_{15}]], \\ M_{13} &= mgb1(s_1, s_3, \{(s_1, s_3), (s_2, s_4)\}), \\ M_{15} &= mgb1(s_1, s_5, \{(s_1, s_3), (s_2, s_4)\}) = false. \end{aligned}$$

Since $mgb1(s_1, s_3, \{(s_1, s_3), (s_2, s_4)\}) = true$, the mgb after simplification is
$mgb(s_1, s_3) \equiv [x = y] \wedge [1 \le x \le 2]$. □

5 UNTIMED BISIMULATION EQUIVALENCE AND ITS VERIFICATION

Definition 4 An *untimed bisimulation relation* R is a binary relation on a set of instances of TSLTS states $\{\rho(s) | s\text{:a TSLTS state}, \rho\text{:an assignment}\}$, which satisfies the following conditions:

- R is a symmetric relation, and
- if $(\rho_i(s_i), \rho_j(s_j)) \in R$, then all of the following conditions hold:

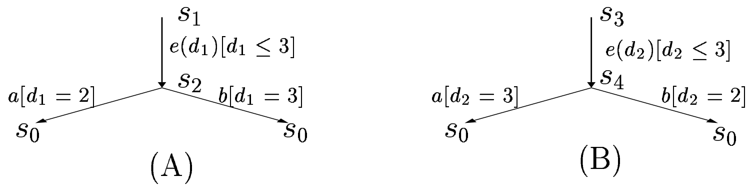

Figure 4 Example of A-TSLTS where s_1 and s_3 are not untimed bisimulation equivalent.

- For any time value t, if $\rho_i(s_i) \xrightarrow{t} \rho'_i(s'_i)$, then there exist some s'_j, ρ'_j and some time value t' such that $\rho_j(s_j) \xrightarrow{t'} \rho'_j(s'_j)$ and $(\rho'_i(s'_i), \rho'_j(s'_j)) \in R$,
- For any action name a in the TSLTS, if $\rho_i(s_i) \overset{a}{\Longrightarrow} \rho'_i(s'_i)$, then there exist some s'_j and ρ'_j such that $\rho_j(s_j) \overset{a}{\Longrightarrow} \rho'_j(s'_j)$ and $(\rho'_i(s'_i), \rho'_j(s'_j)) \in R$. Here $\overset{a}{\Longrightarrow} \overset{\text{def}}{=} \xrightarrow{t_1}\xrightarrow{a}\xrightarrow{t_2}$ for some time values t_1 and t_2.

If there exists some untimed bisimulation equivalence R such that $(\rho_i(s_i), \rho_j(s_j)) \in R$, the two instances $\rho_i(s_i)$ and $\rho_j(s_j)$ are called *untimed bisimulation equivalent*, which is denoted by $\rho_i(s_i) \sim_u \rho_j(s_j)$. □

The result of the previous section can be extended to untimed bisimulation equivalence. To do this, we have only to modify functions *match_delay*() and *match_action*() to make durations not necessarily be equal when we match delay transitions.

The mgb of idle states is easily expressed by the following formula:

$\forall d[P_i\{d/d_i\} \Rightarrow \exists d'[P_j\{d'/d_j\} \wedge M_{i',j'}]] \wedge \forall d'[P_j\{d'/d_j\} \Rightarrow \exists d[P_i\{d/d_i\} \wedge M_{i',j'}]]$

where $M_{i',j'}$ is the mgb of the next pair of active states.

On the other hand, in order to consider the mgb of the active states for untimed bisimulation equivalence, we must solve the following problem. For the timed bisimulation equivalence, we have only to consider the executable actions at the specified time instant (for example, the action a is executable at time 2, the action b is executable at time 3,. . .). However, it is not the case for the untimed bisimulation equivalence. Consider the two A-TSLTSs in Fig. 4. If we consider the executability of actions at time d only, the states s_1 and s_3 should be untimed equivalent, because for duration $d_1 = 2$ after which only a is executable, there exists a duration $d_2 = 3$ after which only a is also executable, and vice versa. However, for the above example, s_1 and s_3 are not untimed equivalent in the sense of Definition 4, because after the delay of 2.5 units of time, s_1 is in the state such that only b is executable (after more 0.5 units of time elapsed), whereas s_3 is in the state such that only a is executable. So, instead of the executability at the given time instant, we consider the executability at some time after the given time instant. For the above example, when the system is at state s_1 and 2 units of time have elapsed, a is executable now and b is executable after more $3 - 2 = 1$ unit of time elapses. In this case, s_1 is in the state such that both a and b are executable at some time in the future (see Fig. 5-(b)). In general, when d units of time have elapsed and a is executable after more $d' - d$ units of time elapse, i.e., d' satisfies both $d \le d'$ and the transition condition of a, a is executable at some time in the future.

Because of the reasons above, we must loose the executability condition of actions in order to define the mgb of the untimed bisimulation equivalence. So we define that for a given duration d, an action is executable if and only if there exists some duration d' such

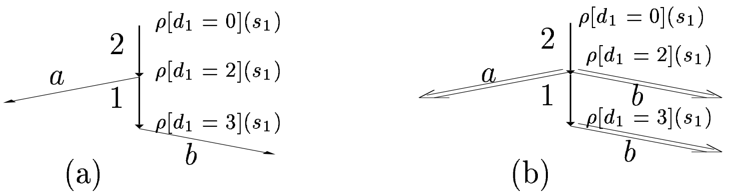

Figure 5 Illustration of (a) timed semantics and (b) untimed semantics of Fig. 4-(A).

that $d \leq d'$ and d' satisfies the transition condition of the action. Note that d', as well as d, must also satisfy the transition condition of the delay transition. Formally, let P denote the transition condition of an action a. Then we say that the action a is *untimedly executable after duration* d, if and only if $\exists d'[d \leq d' \wedge P\{d'/d\}]$. We refer to the condition as the *untimed transition condition*. Since we frequently consider the predicate of the form $\exists d'[d \leq d' \wedge P\{d'/d\}]$ w.r.t. P and the variable d, we abbreviate it as $\mathcal{F}_d P$. Note that if the transition condition of the most recent delay transition is D, then d ranges over the solutions of D. However, $\mathcal{F}_d P$ may have a solution d' which does not satisfy D, which is incorrect. (Consider the untimed transition condition of b in the sequence $s_0 \xrightarrow{e(d),d\leq 2} s_1 \xrightarrow{b,d=3} s_2$.) In this case, the untimed transition condition becomes $\mathcal{F}_d\{P \wedge D\}$.

Using the untimed transition conditions, the mgb of the active states (s_i, s_j) for untimed case is given as follows. Firstly, if s_i and s_j are untimed bisimulation equivalent w.r.t. an assignment ρ, for any action a in a set *Act* of all actions, the following condition holds. Suppose that the most recent delay transitions of s_i and s_j are $s_{i_0} \xrightarrow{e(d_i),D_i} s_i$ for some s_{i_0}, and $s_{j_0} \xrightarrow{e(d_j),D_j} s_j$ for some s_{j_0}, respectively. Note that the delay variable d_i (d_j) ranges over solutions of the predicate D_i (D_j, respectively). For any possible transition $s_i \xrightarrow{a,P_k} s_{i_k}$ whose untimed transition condition $\mathcal{F}_{d_i}[P_k \wedge D_i]$ satisfies $\rho \models \mathcal{F}_{d_i}[P_k \wedge D_i]$, if the action a is untimedly executable, there must exist some transition $s_j \xrightarrow{a,Q_l} s_{j_l}$ whose untimed transition condition $\mathcal{F}_{d_j}[Q_l \wedge D_j]$ also satisfies $\rho \models \mathcal{F}_{d_j}[Q_l \wedge D_j]$ and the destinations s_{i_k} and s_{j_l} must be untimed bisimulation equivalent w.r.t. $\rho[d_i \rightarrow d_i', d_j \rightarrow d_j']$ ($\rho[d_i \rightarrow d_i', d_j \rightarrow d_j']$ must satisfy the mgb for (s_{i_k}, s_{j_l})). Here $\rho[d_i \rightarrow d_i', d_j \rightarrow d_j']$ denotes the same assignment as ρ except the names of variables d_i and d_j are replaced with d_i' and d_j', respectively. Note that since it is assumed that a is untimedly executed, the executed time of a at the state s_i is not d_i but d_i'. So the destinations s_{i_k} and s_{j_l} can be reached with the values of not d_i and d_j but d_i' and d_j'. That is why s_{i_k} and s_{j_l} must be untimed equivalent w.r.t. $\rho[d_i \rightarrow d_i', d_j \rightarrow d_j']$. The above discussions must be true when s_i and s_j are exchanged. Therefore, similar to the timed case, we obtain the mgb of active states s_i and s_j for untimed bisimulation equivalence as:

$$\bigwedge_{k\in K}\{\mathcal{F}_{d_i}[P_k \wedge D_i \Rightarrow \bigvee_{l\in L}\{\mathcal{F}_{d_j}[Q_l \wedge D_j \wedge M_{k,l}]\}]\}$$
$$\wedge \bigwedge_{l\in L}\{\mathcal{F}_{d_j}[Q_l \wedge D_j \Rightarrow \bigvee_{k\in K}\{\mathcal{F}_{d_i}[P_k \wedge D_i \wedge M_{k,l}]\}]\}$$

where, $K = \{k | s_i \xrightarrow{a,P_k} s_{i_k}\}$, $L = \{l | s_j \xrightarrow{a,Q_l} s_{j_l}\}$, $M_{k,l} = mgb(s_{i_k}, s_{j_l})$, $s_{i_0} \xrightarrow{e(d_i),D_i} s_i$ for some s_{i_0}, and $s_{j_0} \xrightarrow{e(d_j),D_j} s_j$ for some s_{j_0}.

The functions *match_delay*() and *match_action*() can be modified properly for untimed bisimulation equivalence according to the mgb obtained above. Although, we omit the detailed definition for lack of space.

Example 4 Consider the two A-TSLTSs in Fig 4. The mgb of (s_1, s_3) for the untimed bisimulation equivalence can be obtained as follows:

$$mgb(s_1, s_3) \quad = \quad \forall d_1[d_1 \leq 3 \Rightarrow \exists d_2[d_2 \leq 3 \wedge M_{24}]] \wedge \forall d_2[d_2 \leq 3 \Rightarrow \exists d_1[d_1 \leq 3 \wedge M_{24}]],$$

where,

$$\begin{aligned} M_{24} \quad = \quad & \exists d_1'[d_1 \leq d_1' \wedge d_1' \leq 3 \wedge d_1' = 2 \Rightarrow \exists d_2'[d_2 \leq d_2' \wedge d_2' \leq 3 \wedge d_2' = 3 \wedge M_{00}]] \\ & \wedge \exists d_2'[d_2 \leq d_2' \wedge d_2' \leq 3 \wedge d_2' = 3 \Rightarrow \exists d_1'[d_1 \leq d_1' \wedge d_1' \leq 3 \wedge d_1' = 2 \wedge M_{00}]] \\ & \wedge \exists d_1'[d_1 \leq d_1' \wedge d_1' \leq 3 \wedge d_1' = 3 \Rightarrow \exists d_2'[d_2 \leq d_2' \wedge d_2' \leq 3 \wedge d_2' = 2 \wedge M_{00}]] \\ & \wedge \exists d_2'[d_2 \leq d_2' \wedge d_2' \leq 3 \wedge d_2' = 2 \Rightarrow \exists d_1'[d_1 \leq d_1' \wedge d_1' \leq 3 \wedge d_1' = 3 \wedge M_{00}]], \\ M_{00} \quad = \quad & true. \end{aligned}$$

After simplifying the above formula, we obtain $M_{24} \equiv (d_1 \leq 2 \wedge d_2 \leq 2) \vee (d_1 > 3 \wedge d_2 > 3)$. So we get $mgb(s_1, s_3) \equiv false$, that is, s_1 and s_3 are not untimed bisimulation equivalent. □

6 AN APPLICATION TO A TIMED LOTOS

The verification method proposed so far should also apply for any structural process description languages, if we could give a correctness-preserving transformation from the expression of the language into A-TSLTS. In this section, we apply our method for the language LOTOS/T, a formal description language of timed processes proposed in (Nakata et al. 1994).

6.1 LOTOS/T

For readers' convenience, here we give a brief summary of the language LOTOS/T.

Definition 5 Behaviour expressions of LOTOS/T is the same as LOTOS, except it has a timed action prefix $a[P(t, \bar{x})]; B$ to describe timing constraint of an action. Here, $P(t, \bar{x})$ stands for a Presburger formula(Hopcroft and Ullman 1979), that is, a first order logic formula whose atoms are integer linear inequalities, which has a free variable t and other free variables x_i. Here $\bar{x} \overset{\text{def}}{=} (x_1, x_2, \ldots, x_j)$ for some j. Intuitively, t represents the current time, e_i stands for an integer linear expression and $\bar{e} \overset{\text{def}}{=} (e_1, e_2, \ldots, e_k)$ for some k, where each e_i is an integer linear expression. □

In LOTOS/T, time constraints of actions are described in a subclass of Presburger formulas, more specifically, logical combinations of the atoms each of which takes the form of either $e_l \leq t$, $t \leq e_u$ or $x = t$. Here, e_l (e_u) is an integer linear expression representing the lower bound (upper bound, respectively) of the time an action is executable. The atomic formula $x = t$ means that the action's executed time is assigned to the variable x. For simplicity, we use an abbreviation $e_l \leq t \leq e_u$ for $e_l \leq t \wedge t \leq e_u$. Other symbols of inequality such as $<$,$\geq$,etc. may also be used. In our semantics, an upper bound e_u specified as a time constraint of an action means that the action *must* be executed no later than e_u. In this case, we say that *urgency* of the action at time e_u is specified. Note that in our language, executability and urgency of each action at each given time t are decidable(Nakata et al. 1994).

Example 5 $\quad B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0 + 3]; c[t = x_0 + 4]; \textbf{stop}$

The behaviour expression B represents the following behaviour. The action a must be executed between time 2 and 3, and the execution time of a is assigned to the variable x_0. Then b must be executed exactly 3 units of time after the execution of a. And then c must be executed exactly 4 units of time after a. $\square$

6.2 Mapping LOTOS/T into A-TSLTS

In (Nakata et al. 1994), a structured operational semantics of LOTOS/T expressions on a discrete (integer) time domain is defined. Our intention is to define another structured operational semantics of LOTOS/T which maps a LOTOS/T expression to an A-TSLTS. In the latter semantics, it does not matter which time domain is considered. To achieve this, firstly we define each state of the obtained A-TSLTS corresponds to an expression of LOTOS/T with a *mark* 'i' or 'a'. The mark indicates which category of states (idle or active) the state itself belongs to (Note that for one behaviour expression B, *both* of two states (B, i) and (B, a) are introduced). Secondly, for each idle state (B, i), where B is a LOTOS/T expression, we define an delay transition starting with (B, i) by inference rules. Finally, for each active state (B, a), we define an action transition starting with (B, a), and an entire inference system which derives A-TSLTSs from LOTOS/T expressions is given. Please note that we simply define $DVar((B, i))$ (or $DVar((B, a))$) as $DVar(B)$, where $DVar(B)$ represents the set of all defined (free) variables in B. Informally, a defined variable means the variable whose value has been already determined by previous execution of actions. For example, w.r.t. the behaviour expression $a[x = t]; b[t \leq x + 3]; \mathbf{stop}$, the variable x in the subexpression $b[t \leq x+3]; \mathbf{stop}$ is a defined variable because the executed time of a has been assigned to x. On the other hand, x in $a[x = t]; b[t \leq x+3]; \mathbf{stop}$ is not a defined variable, since the value of the variable x has not been assigned at this moment. The formal definition of defined variables appears in (Nakata et al. 1994). In the rest of the paper, we assume $DVar(B)$ is a set of defined variables of the subexpression B w.r.t. the entire behaviour expression. Since it is always obvious which expression we assume as the entire behaviour expression (we always assume it is the behaviour expression of the initial state), we simply refer to the defined variables of the expression B as $DVar(B)$.

Delay Transitions of LOTOS/T

Basically, a delay transition from an idle state (B, i) is defined as follows.

- A new delay variable d, which is not used by the behaviour expression B, is introduced to represent the duration of the delay transition.
- The transition condition is defined so that it exactly expresses the possible range of delay of the behaviour expression B.
- The destination state (B', a) of the transition is defined so that it represents the behaviour after d units of time has elapsed. The behaviour expression B' may contain the variable d because the following behaviour might depend on how much time elapsed on this delay transition.

For example, consider a behaviour expression $B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0+3]; c[t = x_0+4]; \mathbf{stop}$. From the definition of LOTOS/T, up to 3 units of time of delay are possible from the idle state (B, i). A delay variable d is introduced to represent the duration. Then, the transition condition of the outgoing delay transition of (B, i) is defined as '$d \leq 3$'.

To consider the state where d units of time have elapsed, every occurrence of t in B is replaced with $(t+d)$. This is the extension of (Nakata et al. 1994). So, the delay transition from (B,i) can be defined as

$$(B,i) \overset{e(d),d\leq 3}{\longrightarrow} (a[2 \leq (t+d) \leq 3 \wedge x_0 = (t+d)]; b[(t+d) = x_0 + 3]; \\ c[(t+d) = x_0 + 4]; \mathbf{stop}, a).$$

The condition such as $d \leq 3$ is easily obtained from $[2 \leq t \leq 3 \wedge x_0 = t]$the transition condition of a. In this case, $\exists d' \exists x_0 [d \leq d' \wedge [2 \leq d' \leq 3 \wedge x_0 = d']]$ is equivalent to $d \leq 3$. In general, if $B = a[P(t,x)]; B'$, then the delay transition from (B,i) is defined as $(B,i) \overset{e(d),\exists d' \exists x[d \leq d' \wedge P(d',x)]}{\longrightarrow} (B\{(t+d)/t\}, a)$.

Action Transitions of LOTOS/T

From the previous section, each active state (B,a), which is reachable from any idle state by a delay transition, represent the behaviour where d units of time have elapsed. So, similar to (Nakata et al. 1994), the transition condition is defined as the condition whether the first action is executable at time 0. The transition condition may contain some undefined variable to which the executed time of the action will be assigned.

Since the action is considered instantaneous, we do not have to consider delay in action transition. So the destination behaviour is obtained similarly to LOTOS.

For example, for the behaviour expression

$$B'' = a[2 \leq (t+d) \leq 3 \wedge x = (t+d)]; b[(t+d) = x+3]; c[(t+d) = x+4]; \mathbf{stop},$$

the action transition

$$(B'', a) \overset{a,2\leq(0+d)\leq 3\wedge x=(0+d)}{\longrightarrow} (b[(t+d) = x+3]; c[(t+d) = x+4]; \mathbf{stop}, i)$$

is defined. Note that since $[2 \leq (0+d) \leq 3 \wedge x = (0+d)]$ holds after a is executed, the value of x is equal to the duration d in the succeeding behaviour.

When a process is defined recursively such as $P(x) := a[t \leq x+3 \wedge y = t]; b[t \leq y \wedge z = t]; P(z)$, the states $(P(x), i)$ and $(P(z), i)$ are essentially the same state if $x = z$. However, because the names of the variables are different, the two states are treated differently in the symbolic semantics. To unify the two states above, we replace these variables with the *minimum* one of all possible new variables (we assume some total order is defined on variables). This is similar to (Jonsson and Parrow 1989).

For example, assume that the set of variables is $\{x, y, z, d, d'\}$ and that a total order of the variables is defined as $x < y < z < d < d'$. For the above example $P(x)$, the corresponding A-TSLTS is obtained as follows.

$$(P(x), i) \overset{e(d),d\leq x+3}{\longrightarrow} (a[(t+d) \leq x+3 \wedge y = (t+d)]; b[(t+d) \leq y \wedge z = (t+d)]; P(z), a)$$

$$(a[(t+d) \leq x+3 \wedge y = (t+d)]; b[(t+d) \leq y \wedge z = (t+d)]; P(z), a) \\ \overset{a,(0+d)\leq x+3\wedge \underline{y}=(0+d)}{\longrightarrow} (b[(t+d) \leq \underline{y} \wedge z = (t+d)]; P(z), i)$$

The undefined variable y is replaced with y itself. Since the defined variable x is also contained in the condition, the minimum new variable is y.

$$(b[(t+d) \leq y \wedge z = (t+d)]; P(y), i) \overset{e(d'),d'+d\leq y}{\longrightarrow} (b[(t+d+d') \leq y \wedge z = (t+d+d')];$$

$P(z), a)$

$(b[(t+d+d') \leq y \wedge z = (t+d+d')]; P(z), a) \xrightarrow{b,(0+d+d') \leq y \wedge \underline{x} = (0+d+d')} (P(\underline{x}), i)$

The undefined variable z is replaced with the minimum new variable x. Because of the replacement, the obtained A-TSLTS has a loop, which corresponds to the recursion.

Nondeterministic Choice and Parallel Execution

For all compositional operators of LOTOS/T, delay and action transitions are also defined. Because of the space limitations, we only describe how transitions of choice and parallel constructs are defined. The other cases are similar.

For choice constructs $B_1[]B_2$, time passing is allowed if and only if it is allowed by *either* B_1 or B_2 (non-persistent choice). This means that time may elapse until reaching the deadline of the first action of either B_1 or B_2. [*] So, in general, the delay transition can be defined as

$(B_1[]B_2, i) \xrightarrow{e(d), P_1 \vee P_2} (B_1'[]B_2', a)$,

if $(B_1, i) \xrightarrow{e(d), P_1} (B_1', a)$ and $(B_2, i) \xrightarrow{e(d), P_2} (B_2', a)$. The action transitions are defined similar to LOTOS.

For example, if $B_1 = a[t \leq 2]; \textbf{stop}$ and $B_2 = b[t \leq 3]; \textbf{stop}$, then

$(B_1[]B_2, i) \xrightarrow{e(d), d \leq 2 \vee d \leq 3} (a[(t+d) \leq 2]; \textbf{stop}[]b[(t+d) \leq 3]; \textbf{stop}, a)$,

$(a[(t+d) \leq 2]; \textbf{stop}[]b[(t+d) \leq 3]; \textbf{stop}, a) \xrightarrow{a, (0+d) \leq 2} (\textbf{stop}, i)$,

$(a[(t+d) \leq 2]; \textbf{stop}[]b[(t+d) \leq 3]; \textbf{stop}, a) \xrightarrow{b, (0+d) \leq 3} (\textbf{stop}, i)$.

For parallel constructs $B_1|[G]|B_2$, time of both B_1 and B_2 synchronizes each other in our semantics. In this case, the delay transition from $(B_1|[G]|B_2, i)$ is defined as

$(B_1|[G]|B_2, i) \xrightarrow{e(d), P_1 \wedge P_2} (B_1'|[G]|B_2', a)$,

where $(B_1, i) \xrightarrow{e(d), P_1} (B_1', a)$ and $(B_2, i) \xrightarrow{e(d), P_2} (B_2', a)$. The case of synchronized action transition (the case where the action a is in G) is similar. That is, the transition condition is a logical product of the transition condition of each component. The case of interleaving action transition is similar to LOTOS.

7 CONCLUSIONS

In this paper, we proposed a model A-TSLTS which can describe timed processes, and a verification method of timed verification equivalence for an A-TSLTS using a similar method as (Hennessy and Lin 1995). We also presented how the result can be applied to structural process description languages such as LOTOS/T.

In contrast to other proposals for timed processes, our model allows arbitrary decidable 1st-order logic on any time domain for describing time constraints. In the model we can

[*]Note that if we use persistent choice semantics instead, we have only to modify the guard predicate of the delay transition from $P_1 \vee P_2$ to $P_1 \wedge P_2$ (i.e., time passing is allowed as long as it is allowed *both* B_1 and B_2).

describe time constraints in a very flexible way and still we can verify timed bisimulation equivalence whose cost is independent of the absolute value of constants used in the time constraints. Although we do not handle value-passing in this paper, our model can easily be extended to the model with both time and value-passing by extending action transitions to have input/output values.

In our method, the verification of equivalence for LOTOS/T expressions is decidable because of the decidability of Presburger Arithmetics. Our research group has implemented an efficient decision procedure of Presburger Arithmetics(Kitamichi et al. 1994).

Our model has an expressive power of describing timing constraints by 1st-order formulas which contains some quantifiers. This might be too powerful for some applications. However, in (Nakata et al. 1995), we have proposed a protocol synthesis method for LOTOS/T service specifications. In the method, derived protocol entity specifications generally contains existential quantifiers to express complicated timing dependencies among actions executed at different nodes. In such an application, our method becomes useful.

Our method is only applicable to the finite state A-TSLTS. For proving equivalence of non-finite-state timed processes, some axiomatic approaches such as (Fokkink and Klusener 1995) have been proposed. However, since untimed bisimulation equivalence is not congruence and the weakest congruence stronger than untimed bisimulation equivalence is equivalent to timed bisimulation equivalence(Larsen and Wang 1993; Alur et al. 1994), an axiomatic approach for proving untimed bisimulation equivalence is unlikely. In this case, our method is still useful.

The question whether (time deterministic) A-TSLTSs are as expressive as (time deterministic) TSLTSs (modulo timed bisimulation equivalence) is left open.

The future works are to extend the result to the verification of timed weak bisimulation equivalence (internal actions are considered), and to implement the algorithm and evaluate the cost of the verification for practically large processes.

REFERENCES

Alur, R., C. Courcoubetis, and T. A. Henzinger (1994). The observational power of clocks. In *Proc. of CONCUR'94*, Volume 836 of *Lecture Notes in Computer Science*, pp. 162–177. Springer-Verlag.

Bolognesi, T. and F. Lucidi (1992). LOTOS-like process algebras with urgent or timed interactions. In *Formal Description Techniques, IV*, pp. 249–264. IFIP: Elsevier Science Publishers B.V.(North-Holland).

Čerāns, K. (1992). Decidability of bisimulation equivalence for parallel timer processes. In *Proc. of 4th CAV*, Volume 663 of *Lecture Notes in Computer Science*, pp. 302–315. Springer-Verlag.

Chen, L. (1992). An interleaving model for real-time systems. In *Proc. of 2nd Int'l Symp. on Logical Foundations of Computer Science (LFCS'92)*, Volume 620 of *Lecture Notes in Computer Science*, pp. 81–92. Springer-Verlag.

Fokkink, W. and A. Klusener (1995). An effective axiomatization for real time ACP. Report CS-R9542, CWI, Amsterdam. To appear in *Information and Computation*.

Hansson, H. A. (1991). *Time and Probability in Formal Design of Distributed Systems*. Ph.D thesis DoCS 91/27, Dept. of Computer Systems, Uppsala University.

Hennessy, M. and M. Lin (1995). Symbolic bisimulations. *Theoret. Comput. Sci. 138*, 353–389.

Holmer, U., K. Larsen, and Y. Wang (1991). Deciding properties of regular timed processes. In *Proc. of 3rd CAV*, Volume 575 of *Lecture Notes in Computer Science*, pp. 443–453. Springer-Verlag.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley.

ISO (1989). *Information Processing System, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* IS 8807.

Jonsson, B. and J. Parrow (1989). Deciding bisimulation equivalences for a class of non-finite-state programs. In *Proc. of STACS'89*, Volume 349 of *Lecture Notes in Computer Science*, pp. 421–433. Springer-Verlag.

Kitamichi, J., S. Morioka, T. Higashino, and K. Taniguchi (1994). Automatic correctness proof of implementation of synchronous sequential circuits using algebraic approach. In *Proc. of 2nd Int'l Conf. on Theorem Provers in Circuit Design (TPCD'94)*, Volume 901 of *Lecture Notes in Computer Science*, pp. 165–184. Springer-Verlag.

Larsen, K. G. and Y. Wang (1993). Time abstracted bisimulation: Implicit specifications and decidability. In *Proc. of 9th Int'l Conf. on Mathematical Foundations of Programming Semantics (MFPS'93)*, Volume 802 of *Lecture Notes in Computer Science*, pp. 160–175. Springer-Verlag.

Moller, F. and C. Tofts (1990). A temporal calculus of communicating systems. In *Proc. of CONCUR '90*, Volume 458 of *Lecture Notes in Computer Science*, pp. 401–415. Springer-Verlag.

Nakata, A., T. Higashino, and K. Taniguchi (1994). LOTOS enhancement to specify time constraints among nonadjacent actions using first order logic. In *Formal Description Techniques, VI (FORTE'93)*, pp. 451–466. IFIP: Elsevier Science Publishers B.V. (North-Holland).

Nakata, A., T. Higashino, and K. Taniguchi (1995). Protocol synthesis from timed and structured specifications. In *Proc. of Int'l Conf. on Network Protocols (ICNP-95)*, pp. 74–81. IEEE: IEEE Computer Society Press.

Wang, Y. (1991). CCS + time = an interleaving model for real time systems. In *Proc. of ICALP '91*, Volume 510 of *Lecture Notes in Computer Science*, pp. 217–228. Springer-Verlag.

Mr. Akio Nakata received B.E. and M.E. degrees in Information and Computer Sciences from Osaka University in 1992 and 1994, respectively. He is now a Ph.D. student of Department of Information and Computer Sciences, Osaka University. His research interests include process algebra, temporal logic, specification and verification of distributed systems, and formal description techniques.

Dr. Teruo Higashino received the B.E., M.E., and Ph.D. degrees in Information and Computer Sciences from Osaka University in 1979, 1981 and 1984, respectively. He is an Associate Professor of the Department of Information and Computer Sciences, Osaka University. His research interests include design and analysis of distributed computing systems, specification and verification of communication protocols, and formal description techniques. He was a Program co-chair of IWPTS'94.

Dr. Kenichi Taniguchi received the B.E., M.E. degrees in electronics engineering and Ph.D. degree in control engineering from Osaka University, Osaka, Japan, in 1965, 1967 and 1970, respectively. He is a Professor of the Department of Information and Computer Sciences, Osaka University. His research interests include design methods for distributed systems and communication protocols and algebraic methods for software development and hardware design and verification.