LOTOS enhancement to specify time constraint among non-adjacent actions using first order logic

Akio Nakata, Teruo Higashino and Kenichi Taniguchi

Department of Information and Computer Science, Osaka University,
1-1 Machikaneyama-cho, Toyonaka-shi, Osaka, 560 Japan

Abstract In this paper, we propose a language LOTOS/T, which is an enhancement of Basic LOTOS. LOTOS/T enables us to describe time constraints in formulas of 1st-order predicate logic. The user only describes the logical relation of time at which each action must be executed. Use of equality ($=$) and inequality ($\leq$) as the time constraints enables us to describe intervals, timeout and delay easily. We define the syntax and semantics of LOTOS/T formally. The semantic model of LOTOS/T is the Labelled Transition System (LTS). We give the inference rules for constructing the LTS's from given LOTOS/T expressions. The LTS's can be constructed mechanically. Then, we show how flexible and convenient to specify practical real-time systems in LOTOS/T with an example. We also define both timed and untimed bisimulation equivalences and give a method to verify their equivalences mechanically.

Keyword Codes: D.3.1; F.4.3; F.3.2
Keywords: Timed LOTOS, Formal Definitions and Theory; First Order Logic, Mathematical Logic; Equivalence, Semantics of Program Languages

1. Introduction

Formal languages based on Process Algebra, such as CCS[1], CSP[2], ACP[3], LOTOS[4] and so on, have been proposed to specify communication protocols and distributed systems. Although these languages can express temporal ordering of the actions, they cannot express explicit time constraints among the actions. It is necessary for the real-time systems and communication protocols to specify discrete quantitative time, because time constraints of such systems are frequently altered depending on implementations. In such cases, we must guarantee that the system's essential behaviour would not be changed.

In the latest years many languages have been proposed to describe real-time properties [5–10]. For example, timed extensions of CCS[5–7], introduced several primitive operators such as delay and timeout operators to describe real-time properties. However, in these languages, even describing a simple time constraint that some

action has to be done within a given time interval yields to a complicated description. Although ACP_ρ[8], based on ACP, can associate any time intervals ranged over reals to any action, there's (currently) no way to prove weak bisimulation equivalence. Moreover, the semantic model of ACP_ρ contains infinite states, which makes it difficult to apply many other verification methods. TIC[9], based on LOTOS, is restricted to specify time constraints between only adjacent two actions. CELOTOS[10] introduced clocks, which can read or reset to zero, to describe time constraints among arbitrary actions. However, CELOTOS cannot specify urgency of actions[1]

In this paper, we propose a language 'LOTOS/T', which improved the issues described above, including the urgency issue. LOTOS/T, which is a timed enhancement of Basic LOTOS, allows us to describe time constraints by the 1st-order predicate logic formulas. The 1st-order predicate logic is well-studied, suitable for automatic verification and makes it easy to describe complicated constraints in 'as is' way. Time is considered discrete. Each process has its own time-table(clock), which is started when it is invoked. Time is expressed as a non-negative integer. The semantic model of LOTOS/T is the Labelled Transition System (LTS) used in LOTOS. Unit time progress is expressed by the action `tic`. We give the inference rules for constructing the LTS's from given LOTOS/T expressions. Time constraints are described by predicates on integers, which must contain a special free variable t (denotes the current time) and may contain other free variables, associated to each action. Use of equality($=$) and inequality ($\leq$) in the predicate will enable us to describe intervals, timeout or delay easily and naturally. Moreover, time at which an action occurred can be assigned to a variable. So it is possible to describe time constraints against actions which are not direct successors. For upward compatibility, if no predicate are associated to the action, the predicate 'true' is assumed for its time constraint. In this case, the action is considered executable at any moment (not urgent). The LTS's can be constructed from given LOTOS/T expressions mechanically using the inference rules.

Two equivalences are introduced, the first is timed (strong/weak) bisimulation equivalence and the last is untimed bisimulation equivalence. Timed bisimulation equivalence is used for checking whether two systems are equivalent and have the same time constraints. Untimed bisimulation equivalence is used for checking whether two systems are equivalent in spite of the different time constraints. If the corresponding LTS's are finite, we can easily check the two bisimulation equivalences by the algorithms in [13, 14].

This paper is organized as follows. In Section 2, the syntax and semantics of LOTOS/T are defined formally. In Section 3, the definition of equivalences related to timed semantics is given. In Section 4, a simple but practical example is provided. Section 5 concludes this paper.

[1]We say that an action is urgent if the action must necessarily be executed at the current time. The urgency issue is mentioned in many papers including [11, 12].

2. Definitions

2.1. Syntax

The syntax of LOTOS/T is defined as follows.

Definition 1 *Behaviour expressions* of LOTOS/T are defined as follows (the priority of operators are analogous to LOTOS):

$E :=$	**stop**	(non-temporal deadlock)
$\mid$	**exit**	(successful termination)
$\mid$	$a; E$	(untimed action prefix)
$\mid$	$a[P(t, \bar{x})]; E$	(time constrained action prefix)
$\mid$	$E[]E$	(choice)
$\mid$	$E\|\|\|E$	(interleaving)
$\mid$	$E\|\|E$	(synchronization)
$\mid$	$E\|[A]\|E$	(generic parallel composition)
$\mid$	$E[> E$	(disabling)
$\mid$	$E >> E$	(enabling)
$\mid$	hide A in E	(hiding)
$\mid$	$P[g_1, \ldots, g_k](\bar{e})$	(process invocation)

where $a \in Act \cup \{i\}$ (Act denotes a finite set of all observable actions, i denotes an internal action) , $A \subset Act$, $k \in \mathbf{N}$ ($\mathbf{N}$ denotes a set of natural numbers), and $P(t, \bar{x})$ stands for a predicate which has a free variable t, denoting the current time, and other variables $\bar{x}$ ($\bar{x}$ denotes a vector of the variables). $\bar{e}$ denotes a vector of the value-expressions.

Predicates are well-formed formulas of 1st-order theory of integers containing $=$, $\leq$ as atomic predicates, $+$, $-$ as functions. Var denotes a set of all variables of the 1st-order theory. Note that this 1st-order theory is decidable because it is, essentially, a subset of Presburger Arithmetics[15]. □

First, we will give an informal explanation of LOTOS/T.

Example 1

$$B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0 + 3]; \mathbf{stop}$$

B denotes a process which executes a between time 2 and 3 and executes b after 3 unit of time elapsed. The predicate $x_0 = t$ denotes that the executing time of a is assigned to the variable x_0. □

The semantic model of LOTOS/T is the LTS. We intend that the LTS in Figure 1 denotes the operational semantics of B.

This LTS is obtained as follows. In Figure 1, the root node corresponds to B. First, only the unit time progress action `tic` is executable for B. Therefore, the edge $\xrightarrow{\texttt{tic}}$ is appended to the root node. If the `tic` is executed, then one unit time elapsed. Since the current time is incremented, $[t + 1/t]B$ is obtained as the new behaviour expression. Here, $[e/x]B$ denotes a behaviour expression B whose every

occurrence of the variable x is replaced with the expression e.

At the state $[t+1/t]B$, only `tic` is executable. Then $[t+1/t]B \xrightarrow{\texttt{tic}} [t+2/t]B$ is appended. At the state $[t+2/t]B$, the `tic` and action a are executable. If the `tic` is executed, then $[t+3/t]B$, that is, $a[2 \leq t+3 \leq 3 \wedge x_0 = t+3]; b[t+3 = x_0+3]; \mathbf{stop}$ is obtained. If the `tic` was executed for $[t+3/t]B$, then the action a could never be executed. In this case, we say that the action a is urgent, that is, the action a must be executed immediately (before the `tic` is executed). Then only a is executable. If a is executed, then "0" is assigned[2] to the variable t in the predicate "$2 \leq t+3 \leq 3 \wedge x_0 = t+3$", which has already been aged by 3 units of time from the initial predicate "$2 \leq t \leq 3 \wedge x_0 = t$" through the operation $[t+3/t]$. Since $x_0 = t+3$, the value of the variable x_0 is fixed to 3, and $b[t+3 = 3+3]; \mathbf{stop}$ is obtained as the new state (behaviour expression). So b is executed after 3 units of time are elapsed.

Next, we will give a formal definition of LOTOS/T. First, we will introduce the notion of the predicate contexts and defined/undefined variables. In order to discuss whether satisfiability of predicates are decidable, we must define which variables have some fixed values and which ones are not. Consider a predicate "$t = x^5 - 7x^3 + 4$". If some value is assigned to x, satisfiability of the predicate is easy to decide for any given values of t. However, if no values are assigned to x, for a given value of t, a 5th-degree equation must be solved to decide satisfiability. So we need a formal definition of whether or not a variable's value is defined. We also need a notion of predicate contexts because the definition of a defined/undefined variable depends on where the variable occurs in a behaviour expression of LOTOS/T.

Definition 2 *Predicate contexts* are syntactically defined by the following BNF. Here, E is the syntactical component representing a behaviour expression which is used in Definition 1.

$$\begin{aligned} C \;:=\; & a[\bullet]; E \mid a[P(t,\bar{x})]; C \\ & \mid C[]E \mid E[]C \mid C|[A]|E \mid E|[A]|C \\ & \mid C|||E \mid E|||C \mid C||E \mid E||C \\ & \mid C[> E \mid E[> C \mid C >> E \mid E >> C. \end{aligned}$$

□

For example, let us consider the behaviour expression B in Example 1. For this behaviour expression B, the following two predicate contexts are possible:

$$C = a[\bullet]; b[t = x_0 + 3]; \mathbf{stop}$$
$$C' = a[2 \leq t \leq 3 \wedge x_0 = t]; b[\bullet]; \mathbf{stop}$$

Here, "$\bullet$" denotes a time constraint of the current action. In the context C, the variable "x_0" is undefined because the value of the variable "x_0" is not fixed until a

[2]Please note that assigning 0 to t in the 3 units of time aged predicate "$2 \leq t+3 \leq 3 \wedge x_0 = t+3$" is equivalent to assigning 3 to the variable t in the initial predicate "$2 \leq t \leq 3 \wedge x_0 = t$." We only have to check satisfiability of the predicate in a case of $t = 0$, because behaviour expressions are properly aged when $\xrightarrow{\texttt{tic}}$ is added, in order to treat current time as 0.

is executed. However, in the context C', the variable x_0 is defined because the value of x_0 has been fixed before b is executed.

Formally, the defined/undefined variable are decided as follows. Here, $DVar(C)$ and $UVar(C)$ denote the sets of defined/undefined variables for a predicate context C, respectively.

Definition 3 For any predicate context C, $DVar(C) \subset Var$ is defined recursively as follows.

$$\begin{aligned} DVar(a[\bullet]; E) &\stackrel{\text{def}}{=} \emptyset \\ DVar(a[P(t, \bar{x})]; C) &\stackrel{\text{def}}{=} \{y | y \text{ is an element of } \bar{x}\} \cup DVar(C) \\ DVar(C \triangle E) &\stackrel{\text{def}}{=} DVar(C) \\ DVar(E \triangle C) &\stackrel{\text{def}}{=} DVar(C) \\ & \quad (\triangle \in \{[], |[A]|, [>, >>\}) \end{aligned}$$

And $UVar(C) \stackrel{\text{def}}{=} Var - DVar(C)$. □

Hereafter, we define the set of predicates $P(t, \bar{x})$ which can be used in the predicate context C. We believe that the class of predicates $Pres(C)$ defined in Definition 4 is reasonably wide, because time interval, whose bounds are expressed in linear expressions, can be written and the executed time of any preceded actions are referred to in the expressions.

Definition 4 A set of predicates allowed to use in the predicate context C, denoted as $Pres(C)$, is defined as a minimum set which satisfies the following conditions:

- "$e_l \leq t \leq e_u$", "$e_l \leq t$" and "$t \leq e_u$" are in $Pres(C)$. Here, e_l and e_u denote arbitrary terms consisting of only integers, the variables in $DVar(C)$, and operators $+$ and $-$. If e_l and e_u are the same, then "$e_l \leq t \leq e_u$" is abbreviated to "$t = e_u$".
- if $P \in Pres(C)$ and $x \notin FVar(P) \cup DVar(C)$, then "$P \wedge (x = t)$" is in $Pres(C)$.
- if $P_1, P_2 \in Pres(C)$ and $FVar(P_1) \cap FVar(P_2) \cap UVar(C) = \emptyset$, then both "$P_1 \vee P_2$" and "$P_1 \wedge P_2$" are in $Pres(C)$.
- if $P \in Pres(C)$ and $FVar(P) \cap UVar(C) = \emptyset$, then "$\neg P$" is in $Pres(C)$.

where $FVar(P)$ denotes a set of all free variables occurred in a predicate P. □

Note that the predicate $P(t, \bar{x})$ may be described as $P(t, \bar{x}_d, \bar{x}_u)$ if necessary, where the 2nd parameter $\bar{x}_d$ denotes a vector of the defined variables in $\bar{x}$ and the 3rd parameter $\bar{x}_u$ denotes a vector of the undefined variables in $\bar{x}$ under C.

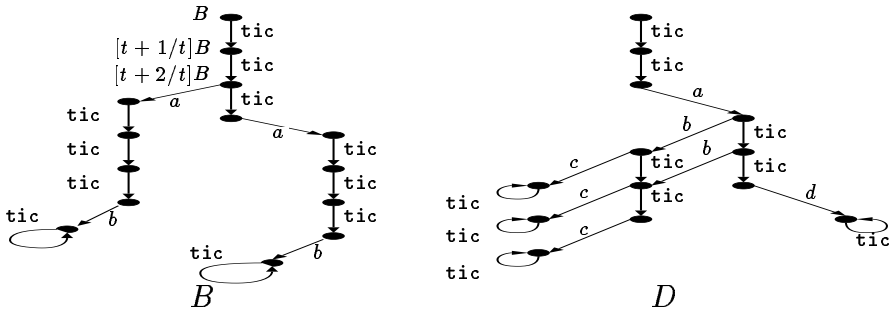

Figure 1. The semantics of B and D

Next, we will explain that $Pres(C)$ defined in Definition 4 is restrictive in spite of its expressive power, that is, satisfiability of a predicate in the class is still decidable. For convenience, we refer to having such a desirable property as *normal*.

First of all, normal predicates are defined.

Definition 5 A predicate $P(t, \bar{x})$ is *normal* under a context C if $P(t, \bar{x})$ satisfies the following conditions.

1. (decidability) For any $n \in \mathbf{N}$ and $\bar{v}$, satisfiabilities of the two formulas $P(n, \bar{v}, \bar{x}_u)$ and $(\mathcal{F}P)(n, \bar{v}) \stackrel{\text{def}}{=} \exists t' \exists \bar{x}_u [t' \geq n \wedge P(t', \bar{v}, \bar{x}_u)]$ are decidable.

2. (uniqueness of substitution) For any $n \in \mathbf{N}$ and $\bar{v}$, there exist unique values $\bar{c}$ such that $P(n, \bar{v}, \bar{c})$ holds if the formula $\exists \bar{x}_u P(n, \bar{v}, \bar{x}_u)$ is satisfiable. Also such values $\bar{c}$ are computable from n and $\bar{v}$ i.e. there exists a partial recursive function $\phi_P(n, \bar{v})$ such that $\exists \bar{x}_u P(n, \bar{v}, \bar{x}_u)$ implies $P(n, \bar{v}, \phi_P(n, \bar{v}))$.

Remark: Condition 1 is needed to make sure that we can construct the semantical model of the expression mechanically. Condition 2 is needed to avoid ambiguity of assigned value to be assigned to variables. □

We say a behaviour expression B is normal iff all predicates appeared in B are normal under its contexts, i.e. for any C and P such that $B = C(P)$, P is normal under C.

For the elements of $Pres(C)$, the following property holds.

Proposition 1 *For any context C, all the predicates in $Pres(C)$ are normal.*

Proof. Since each predicate P in $Pres(C)$ is described as a logical combination of some integer linear inequalities, P and $\mathcal{F}P$ in Definition 5 are expressions in Presburger Arithmetics [15]. Since it is known that satisfiability of Presburger Arithmetics is decidable [15], satisfiabilities of P and $\mathcal{F}P$ are also decidable. Therefore, Condition 1 in Definition 5 holds. Condition 2 also holds. For the details, see [16]. □

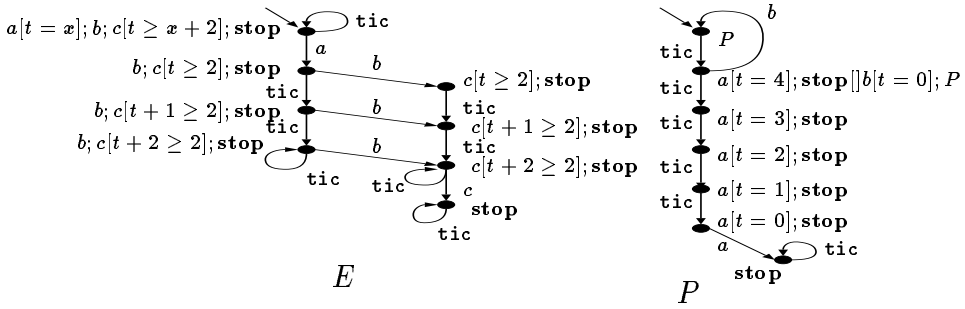

Figure 2. The semantics of E and P

$Pres(C)$ is useful for describing normal predicates. If other class of 1st-order theory is considered, the conditions in Definition 5 does not always hold.

Example 2 Under the predicate context "$a[t=x];b[\bullet];\mathbf{stop}$", "$t=x^2+2x+1\wedge y=2t$" satisfies the conditions 1 and 2 of Definition 5. However, "$t=x^2+2x+1\wedge y>t$" violates the condition 2, and "$t=y^5+9y^2z^3+z^4$" (Diophantine polynomial) violates both. □

Example 3 The following example is also possible. The LTS for D is shown in Figure 1. In this example, time constraints between non-adjacent actions (the actions a and c) are described.

$$D=a[t=2\wedge x_0=t];(b[x_0\leq t\leq x_0+1];c[x_0\leq t\leq x_0+2];\mathbf{stop}\ []d[t=x_0+x_0];\mathbf{stop})$$ □

Example 4 Other examples are given below. The first one contains untimed action sandwiched between time-constrained actions, and infinite interval for the time constraint. The second one describes an infinite behaviour.

1. $E=a[x=t];b;c[t\geq x+2];stop$
2. $P=a[t=5];stop[]b[t=1];P$

The corresponding LTS's are shown in Figure 2. □

In LOTOS/T, untimed or infinite behaviours may be described (for example, the processes E and P in Example 4).

2.2. Operational Semantics

In this section, we will give the formal semantics of LOTOS/T. The operational semantics of LOTOS/T is an extension of LOTOS. The difference is the treatment of transitions of the extra action `tic`. Here we define the operational semantics of LOTOS/T by giving an inference system of the transition relation (see Tables 1 and 2).

Table 1
The inference rules of transition relation: Part 1

Inaction				
$\dfrac{}{\mathbf{stop} \xrightarrow{\mathtt{tic}} \mathbf{stop}}$	(1)		$\dfrac{}{\mathbf{exit} \xrightarrow{\delta} \mathbf{stop}}$	(2)
$\dfrac{}{\mathbf{exit} \xrightarrow{\mathtt{tic}} \mathbf{exit}}$	(3)			
Action Prefix				
$\dfrac{P(0,\bar{c})}{a[P(t,\bar{x})];B \xrightarrow{a} [\bar{c}/\bar{x}]B}$	(4)		$\dfrac{\mathcal{F}P(1)}{a[P(t,\bar{x})];B \xrightarrow{\mathtt{tic}} a[P(t+1,\bar{x})];[t+1/t]B}$	(5)
$\dfrac{}{a;B \xrightarrow{a} B}$	(6)		$\dfrac{}{a;B \xrightarrow{\mathtt{tic}} a;[t+1/t]B}$	(7)
Internal Action				
$\dfrac{P(0,\bar{c})}{i[P(t,\bar{x})];B \xrightarrow{i} [\bar{c}/\bar{x}]B}$	(8)		$\dfrac{\neg P(0,\bar{x}) \quad \mathcal{F}P(1)}{i[P(t,\bar{x})];B \xrightarrow{\mathtt{tic}} i[P(t+1,\bar{x})];[t+1/t]B}$	(9)
$\dfrac{}{i;B \xrightarrow{i} B}$	(10)			
Choice				
$\dfrac{B_1 \xrightarrow{\beta} B_1'}{B_1[]B_2 \xrightarrow{\beta} B_1'}$ iff $\beta \in Act \cup \{\delta, i\}$	(11)		$\dfrac{B_2 \xrightarrow{\beta} B_2'}{B_1[]B_2 \xrightarrow{\beta} B_2'}$ iff $\beta \in Act \cup \{\delta, i\}$	(12)
$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \xrightarrow{\mathtt{tic}} B_2'}{B_1[]B_2 \xrightarrow{\mathtt{tic}} B_1'[]B_2'}$	(13)			
$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \not\xrightarrow{\mathtt{tic}}}{B_1[]B_2 \xrightarrow{\mathtt{tic}} B_1'}$	(14)		$\dfrac{B_2 \xrightarrow{\mathtt{tic}} B_2' \quad B_1 \not\xrightarrow{\mathtt{tic}}}{B_1[]B_2 \xrightarrow{\mathtt{tic}} B_2'}$	(15)

2.2.1. Inaction

The behaviour expression **stop** is extended to express *non-temporally deadlocked process*, which cannot do any other computations except the infinite sequence of `tic`. The behaviour expression **exit** is extended to execute `tic` actions any times before executing δ action.

2.2.2. Action Prefix

The behaviour expression $a[P(t,\bar{x})];B$ means that the action a can occur at time n if $P(n,\bar{c})$ holds for some $\bar{c}$. Because the predicate P is assumed to be normal, satisfiability of $P(n,\bar{x})$ is decidable (from condition 1 of Definition 5), and the value $\bar{c}$ which satisfies $P(n,\bar{c})$ is uniquely computable (from condition 2).

In order to express urgency, we define that the action `tic` cannot occur if the action cannot happen in the future, i.e. $\mathcal{F}P(1) \equiv \exists t' \exists \bar{x}[t' \geq 1 \wedge P(t',\bar{x})]$ does not hold. Satisfiability of $\mathcal{F}P(1)$ is also decidable (from condition 1).

The semantics of the untimed action prefix, $a;B$, is the same as that of $a[true];B$.

2.2.3. Internal Action

For the behaviour expression $i;B$, the internal action is considered always urgent, so its execution is prior to `tic` action. The rest is similar to action prefix.

2.2.4. Choice

We define the choice operator be weak-choice[5], so our choice operator is non-persistent. For example, "$a[t = 1];\mathbf{stop}$" and "$b[t = 2];\mathbf{stop}$" are equivalent to

Table 2
The inference rules of transition relation: Part 2

Parallel	
$\dfrac{B_1 \xrightarrow{\beta} B_1' \quad B_2 \xrightarrow{\beta} B_2'}{B_1\|[A]\|B_2 \xrightarrow{\beta} B_1'\|[A]\|B_2'}$ iff $\beta \in A \cup \{\delta\}$ (16)	$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \xrightarrow{\mathtt{tic}} B_2'}{B_1\|[A]\|B_2 \xrightarrow{\mathtt{tic}} B_1'\|[A]\|B_2'}$ (17)
$\dfrac{B_1 \xrightarrow{a} B_1'}{B_1\|[A]\|B_2 \xrightarrow{a} B_1'\|[A]\|B_2}$ iff $a \notin A \vee a = i$ (18)	$\dfrac{B_2 \xrightarrow{a} B_2'}{B_1\|[A]\|B_2 \xrightarrow{a} B_1\|[A]\|B_2'}$ iff $a \notin A \vee a = i$ (19)
$\dfrac{B_1\|[\emptyset]\|B_2 \xrightarrow{\alpha} B'}{B_1 \|\|\| B_2 \xrightarrow{\alpha} B'}$ iff $\alpha \in Act \cup \{\delta, \mathtt{tic}, i\}$ (20)	$\dfrac{B_1\|[Act]\|B_2 \xrightarrow{\alpha} B'}{B_1 \|\| B_2 \xrightarrow{\alpha} B'}$ iff $\alpha \in Act \cup \{\delta, \mathtt{tic}, i\}$ (21)
Disable	
$\dfrac{B_1 \xrightarrow{a} B_1'}{B_1[> B_2 \xrightarrow{a} B_1'[> B_2}$ (22)	$\dfrac{B_2 \xrightarrow{\beta} B_2'}{B_1[> B_2 \xrightarrow{\beta} B_2'}$ iff $\beta \in Act \cup \{\delta, i\}$ (23)
$\dfrac{B_1 \xrightarrow{\delta} B_1'}{B_1[> B_2 \xrightarrow{\delta} B_1'}$ (24)	$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \xrightarrow{\mathtt{tic}} B_2'}{B_1[> B_2 \xrightarrow{\mathtt{tic}} B_1'[> B_2'}$ (25)
Enable	
$\dfrac{B_1 \xrightarrow{a} B_1'}{B_1 >> B_2 \xrightarrow{a} B_1' >> B_2}$ (26)	$\dfrac{B_1 \xrightarrow{\delta} B_1'}{B_1 >> B_2 \xrightarrow{i} B_2}$ (27)
$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \xrightarrow{\mathtt{tic}} B_2' \quad B_1 \not\xrightarrow{\delta}}{B_1 >> B_2 \xrightarrow{\mathtt{tic}} B_1' >> B_2'}$ (28)	
Hide	
$\dfrac{B \xrightarrow{\beta} B'}{\text{hide } A \text{ in } B \xrightarrow{\beta} \text{hide } A \text{ in } B'}$ iff $\beta \in (Act - A) \cup \{\delta, i\}$ (29)	
$\dfrac{B \xrightarrow{a} B'}{\text{hide } A \text{ in } B \xrightarrow{i} \text{hide } A \text{ in } B'}$ iff $a \in A$ (30)	$\dfrac{B \xrightarrow{\mathtt{tic}} B' \quad B \not\xrightarrow{a} \text{ for all } a \in A}{\text{hide } A \text{ in } B \xrightarrow{\mathtt{tic}} \text{hide } A \text{ in } B'}$ (31)
Process Invocation	
$\dfrac{[\bar{e}/\bar{x}]B\{g_1'/g_1, \ldots, g_k'/g_k\} \xrightarrow{\alpha} B'}{P[g_1', \ldots, g_k'](\bar{e}) \xrightarrow{\alpha} B'}$ iff $\alpha \in Act \cup \{\mathtt{tic}, \delta, i\}$ and $P[g_1, \ldots, g_k](\bar{x}) := B$ is a definition (32)	

"$\texttt{tic}; a; \mathbf{stop}$" and "$\texttt{tic}; \texttt{tic}; b; \mathbf{stop}$", respectively. However, "$a[t = 1]; \mathbf{stop}[]b[t = 2]; \mathbf{stop}$" is not equivalent to "$\texttt{tic}; a; \mathbf{stop}[]\texttt{tic}; \texttt{tic}; b; \mathbf{stop}$" because the choice is occurred at time 0. It must be equivalent to "$\texttt{tic}; (a; \mathbf{stop}[]\texttt{tic}; b; \mathbf{stop})$". The inference rules in Table 1 are introduced to construct the latter semantic model.

2.2.5. Parallel

Parallel operators ($|||$, $||$, $|[A]|$) always synchronize `tic` actions in LOTOS/T. Consequently, the time constraint of interaction is the logical product of the time constraints of the actions in both processes.

ex.) In $a; b[2 \leq t \leq 4]; \mathbf{stop}|[b]|c; b[3 \leq t \leq 5]; \mathbf{stop}$, the time constraint of the interaction b is $3 \leq t \leq 4$.

2.2.6. Disable

The definition is similar to LOTOS except the `tic` action.

2.2.7. Enable

Similar to LOTOS, except `tic` synchronizes unconditionally and enabling is prior to the `tic` action.

2.2.8. Hide

Similar to LOTOS, except `tic` occurs only if B cannot execute the hidden action in order to express urgency of it.

2.2.9. Process Invocation

A process invocation behaves exactly the same as the behaviour at time 0, no matter when it is invoked.

2.3. Consistency of the inference system

It is very important to notice that our inference system, used for defining operational semantics, are consistent. An inference system is called consistent if the existence of a transition is never deduced from the non-existence of the transition itself. If an inference system is inconsistent, the semantic model cannot exist. Unfortunately, our inference system contains negative premises in some inference rules. So consistency is not self-evident. However, our inference rules can be proved consistent by using the *stratification* technique described in [17]. We omit the proof for lack of space.

2.4. Example of LTS construction

By applying the inference rules shown in this section, we can construct the corresponding LTS as follows. Let us consider the process E in Figure 2.

- $E = a[t = x]; b; c[t \geq x + 2]; \mathbf{stop} \xrightarrow{a} b; c[t \geq 2]; \mathbf{stop}$ (by rule (4)),
- $b; c[t \geq 2]; \mathbf{stop} \xrightarrow{\texttt{tic}} b; c[t + 1 \geq 2]; \mathbf{stop}$ (by rule (5)),

and so on.

For the process P in Figure 2, the following actions are possible:

- $P \xrightarrow{\texttt{tic}} a[t=4];\mathbf{stop}[]b[t=0];P$ (by rules (32), (13), (5)),
- $a[t=4];\mathbf{stop}[]b[t=0];P \xrightarrow{\texttt{tic}} a[t=3];\mathbf{stop}$ (by rules (14) and (5)),

and so on.

Note that we regard two states as the same if satisfiability of the corresponding predicates for each t on $0 \leq t < \infty$ are equivalent. For instance, w.r.t. E in Figure 2, $a[t=x];b;c[t \geq x+2];\mathbf{stop} \xrightarrow{\texttt{tic}} a[t+1=x];b;c[t+1 \geq x+2];\mathbf{stop}$ holds by the inference rules. Here, satisfiabilities of two predicates of the action a, $t=x$ and $t+1=x$, are equivalent, i.e.

$$\forall t[0 \leq t \Rightarrow \exists x[t=x] \equiv \exists x'[t+1=x']] \tag{33}$$

holds (Note that x in "$t=x$" and x in "$t+1=x$" have no longer the same value. So we describe the latter formula as "$t+1=x'$").

Furthermore, for any value assignment of x and x', satisfying (33), into two predicate two predicates of the action c,

$$\forall t'[0 \leq t' \Rightarrow [t' \geq x+2] \equiv [t'+1 \geq x'+2]] \tag{34}$$

holds.

To summarize the idea above, we can verify whether E and $[t+1/t]E$ are representing the same state by checking satisfiability of the following predicate:

$$\begin{aligned}\forall t_1[0 \leq t_1 \Rightarrow [\exists x(t_1=x) \equiv \exists x'(t_1+1=x')] \wedge \\ \forall x \forall x'[(t_1=x) \wedge (t_1+1=x') \Rightarrow \\ \forall t_2[0 \leq t_2 \Rightarrow [(t_2 \geq x+2) \equiv (t_2+1 \geq x'+2)]]]]\end{aligned} \tag{35}$$

So we can state $a[t=x];b;c[t \geq x+2];\mathbf{stop} \xrightarrow{\texttt{tic}} a[t=x];b;c[t \geq x+2];\mathbf{stop}$ (i.e. this node has a self loop of `tic`).

Aging (replacing t with $t+1$) does not have an effect on process invocation, since process name does not have the variable t literally. For example, w.r.t. P in Figure 2, $P \xrightarrow{\texttt{tic}} a[t=4];\mathbf{stop}[]b[t=0];P \xrightarrow{b} P$ holds by the inference rules. So the corresponding LTS has a cycle, as shown in Figure 2.

3. Equivalence

3.1. Timed Bisimulation Equivalence

Definition 6 A relation $\mathcal{R}$ is *timed strong bisimulation* if the following condition holds.

> if $B_1 \mathcal{R} B_2$, then for any $a \in Act \cup \{\delta, \texttt{tic}\}$, the following two conditions hold:

1. if $B_1 \xrightarrow{a} B_1'$, then $\exists B_2'[B_2 \xrightarrow{a} B_2'$ and $B_1' \mathcal{R} B_2']$
2. if $B_2 \xrightarrow{a} B_2'$, then $\exists B_1'[B_1 \xrightarrow{a} B_1'$ and $B_1' \mathcal{R} B_2']$ □

Definition 7 The behaviour expressions B and B' are *timed strong bisimulation equivalent*, denoted by $B \sim_t B'$, iff there exists a timed strong bisimulation $\mathcal{R}$ such that $B \mathcal{R} B'$. □

The equational theory of timed strong bisimulation equivalence including expansion theorem is given in [16].

Timed weak bisimulation equivalence ($\approx_t$), where the internal action i is considered unobservable, can also be defined similarly.

Example 5 The following two behaviour expressions are timed strong bisimulation equivalent:

$$B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0 + 3]; B$$
$$C = a[t = 2]; b[t = 5]; C [] a[t = 3]; b[t = 6]; C$$

3.2. Untimed Bisimulation Equivalence

Here we introduce an *untimed bisimulation equivalence* where `tic` is considered unobservable. Using this equivalence, we can prove whether two timed expressions execute the same observable event sequences. Like timed bisimulation equivalence, untimed bisimulation equivalence has two definitions, one is *untimed strong bisimulation equivalence*, where only `tic` is considered unobservable, and the other is *untimed weak bisimulation equivalence*, where both `tic` and i are considered unobservable.

Definition 8 For each action $a \in (Act \cup \{\delta\} - \{\mathtt{tic}\}) \cup \{\epsilon\}$, the relation $\stackrel{a}{\Longrightarrow}$ over behaviour expressions is defined as follows:

$$B \stackrel{a}{\Longrightarrow} B' \stackrel{\text{def}}{=} \begin{cases} B(\xrightarrow{\mathtt{tic}})^* \xrightarrow{a} (\xrightarrow{\mathtt{tic}})^* B', & \text{if } a \in Act \cup \{\delta\} - \{\mathtt{tic}\} \\ B(\xrightarrow{\mathtt{tic}})^* B' & \text{if } a = \epsilon \end{cases}$$ □

Definition 9 A relation $\mathcal{R}$ is *untimed strong bisimulation* if the following condition holds:

if $B_1 \mathcal{R} B_2$, then for any $a \in (Act \cup \{\delta\} - \{\mathtt{tic}\}) \cup \{\epsilon\}$, the following conditions hold:

1. if $B_1 \stackrel{a}{\Longrightarrow} B_1'$, then $\exists B_2'[B_2 \stackrel{a}{\Longrightarrow} B_2'$ and $B_1' \mathcal{R} B_2']$
2. if $B_2 \stackrel{a}{\Longrightarrow} B_2'$, then $\exists B_1'[B_1 \stackrel{a}{\Longrightarrow} B_1'$ and $B_1' \mathcal{R} B_2']$ □

Definition 10 The behaviour expressions B and B' are *untimed strong bisimulation equivalent*, denoted by $B \sim_u B'$, iff there exists a weak bisimulation $\mathcal{R}$ such that $B \mathcal{R} B'$. □

Untimed weak bisimulation equivalence, denoted by $\approx_u$, can be defined similarly.

Proposition 2 *The behaviour expressions which are timed strong[weak] bisimulation equivalent are untimed strong[weak] bisimulation equivalent, respectively, i.e.:*

$$B \sim_t B' \Rightarrow B \sim_u B'$$
$$B \approx_t B' \Rightarrow B \approx_u B' \qquad \square$$

Proposition 3 $\sim_u$ *is not a congruence, i.e.:*

$$\exists B_1, B_2[(B_1 \sim_u B_2) \wedge (B[]B_1 \not\sim_u B[]B_2)]$$

Proof. Choose $B_1 = a[t = 0]; \mathbf{stop}$, $B_2 = a[t = 2]; \mathbf{stop}$ and $B = b[t = 1]; \mathbf{stop}$. $\square$

Note that from Proposition 3, untimed bisimulation equivalence is hardly suitable for axiomatic proof system.

Example 6 Let B and D denote the following expressions, respectively:

$$B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0 + 3]; \mathbf{stop}$$
$$D = a[t = 2]; \mathbf{stop} ||| b[3 \leq t \leq 5]; \mathbf{stop}$$

Then, B and D are untimed strong bisimulation equivalent because

$$\mathcal{R} = \{([t + k/t]B, [t + l/t]D) | 0 \leq k \leq 3 \wedge 0 \leq l \leq 2\}$$
$$\cup \{(b[t + k = m + 3]; \mathbf{stop}, b[3 \leq t + l \leq 5]; \mathbf{stop}) | 2 \leq m \leq 3 \wedge k \leq m + 3 \wedge 3 \leq l \leq 5\}$$
$$\cup \{(\mathbf{stop}, \mathbf{stop})\}$$

is an untimed strong bisimulation which satisfies $B\mathcal{R}D$. $\square$

In the following Proposition, we mention the decidability of these equivalences.

Proposition 4 *If the corresponding LTS's of both B_1 and B_2 are finite, then all the equivalences defined above are decidable.*

Proof. Analogous to [13, 14]. $\square$

Note that the corresponding LTS of a behaviour expression is not always finite, but if the LTS is finite, then equivalences are decidable from Proposition 4.

4. Example

Here we introduce a more practical example. The example shown in Figure 3 models a remote controller or something that has only one press button for input and executes 4 output actions according to the timing patterns of pressing button. The timing patterns are:

```
ONE_KEY_CONTROLLER[p,r,lc,sc,dc]
     := p[t1p=t];
          (lc[t1p+d1<=t<=t1p+d4];r;ONE_KEY_CONTROLLER
           [] r[t<t1p+d1];
               (p[t<t1p+d2 and t2p=t];
                     (r[t<t2p+d3];dc[t2p+d3<=t<=t1p+d4];ONE_KEY_CONTROLLER
                     [] slc[t2p+d3<=t<=t1p+d4];r;ONE_KEY_CONTROLLER)
                [] sc[t1p+d2<=t<=t1p+d4];exit)
           )

+ variables
  t1p: time when the first press occurred.
  t2p: time when the second press occurred.
+ constants
  d1: threshold for the first short or long click
  d2: timeout for the second click
  d3: threshold for the second short or long click
  d4: required maximum total delay between button press and result action
```

Figure 3. Timed specification of one-key controller

- long click once,
- short click once,
- double short click and
- short click followed by long click.

The second one is used for terminating, while others are continued to be accepted infinitely. Pressing button is modeled by the sequence of the actions `p` (short for 'press') and `r` (short for 'release'). The corresponding output actions are `lc` (short for 'long click'), `sc` (short for 'short click') and `dc` (short for 'double click') and `slc` (short for 'short and long click'). If `d2+d3>d4`, it may cause violation of time constraint (temporal deadlock). And if `d1>=d2`, it may cause second click be lost. So the sound implementation must satisfy `d1<d2` and `d2+d3<=d4`.

This will be checked by constructing the LTS for some values to `d1`,`d2`,`d3` and `d4` satisfying above. In the LTS, the temporally deadlocked state has no outgoing arc including `tic`. Whether or not the behaviour has been modified because of the time constraint is checked by verifying untimed bisimulation equivalence with the untimed specification like Figure 4.

```
UNTIMED_ONE_KEY_CONTROLLER[p,r,lc,sc,dc]
     :=p;(i;lc;r;UNTIMED_ONE_KEY_CONTROLLER
          [] r;(p;(r;dc;UNTIMED_ONE_KEY_CONTROLLER
                   []i;slc;r;UNTIMED_ONE_KEY_CONTROLLER)
                []i;sc;exit)
          )
```

Figure 4. Untimed specification of one-key controller

5. Conclusion

We have proposed a language LOTOS/T, a timed enhancement of Basic LOTOS. LOTOS/T enables us to describe time constraints among actions in a flexible way using formulas of 1st-order theory.

In order to construct the LTS from a given LOTOS/T expression mechanically, we need a decision procedure for Presburger Arithmetics. We have developed the decision procedure[18] on a Sun SparcStation ELC. For the predicates given in this paper as examples, satisfiabilities of the predicates can be decided within one second. Even for more complex predicates such as the logical combinations of ten integer linear inequalities, their satisfiabilities can be decided within a few seconds in most cases. Therefore, LOTOS/T is enough powerful for practical purposes and suitable for mechanical proof method. We have developed LOTOS interpreter[19] and a test system for LOTOS with data parameters[20]. Using these systems, we can construct the LTS from a given LOTOS expression mechanically. Now we have a plan to develop the decision procedure for proving the timed/untimed bisimulation equivalences described in Section 3 by using the above tools.

We did not introduce timing-interaction operator defined in [11]. The strength of this is that locality of specification is preserved, as mentioned in [11] (but differs from Timed-Action LOTOS[11] because urgency is still supported in ours). Urgency of interaction can still be expressed in LOTOS/T by hiding the interaction from outside, but urgency of observable interaction cannot be expressed. So expressive power of LOTOS/T is weaker than Timed-Interaction LOTOS and Timed Petri Nets.

Untimed bisimulation equivalence is introduced in order to consider the two processes, which behave the same but in different time constraints (e.g. in different speed), be equivalent. Similar but more advanced investigations are made for CCS in [21, 22].

REFERENCES

1. R. Milner, *A Calculus of Communicating Systems*, vol. 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
2. C. A. R. Hoare, *Communicating Sequential Processes*. Prentice Hall, 1985.
3. J. A. Bergstra and J. W. Klop, "Algebra of communicating processes with abstraction," *Theoret. Comput. Sci.*, vol. 37, pp. 77–121, 1985.
4. ISO, *LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. IS 8807, Jan. 1989.
5. F. Moller and C. Tofts, "A temporal calculus of communicating systems," in *Proc. of CONCUR '90*, vol. 458 of *Lecture Notes in Computer Science*, pp. 401–415, Springer-Verlag, 1990.
6. Y. Wang, "CCS + time = an interleaving model for real time systems," in *Proc. of ICALP '91*, vol. 510 of *Lecture Notes in Computer Science*, pp. 217–228, Springer-Verlag, 1991.
7. M. Hennessy and T. Regan, "A temporal process algebra," in *Formal Description Techniques, III* , pp. 33–48, IFIP, North-Holland, 1991.
8. J. C. M. Baeten and J. A. Bergstra, "Real time process algebra," *Journal of Formal Aspects of Computing Science*, vol. 3, no. 2, pp. 142–188, 1991.
9. J. Quemada, A. Azcorra, and D. Frutos, "TIC: A timed calculus for LOTOS," in *Formal Description Techniques, II*, pp. 195–209, IFIP, North-Holland, 1990.
10. W. H. P. van Hulzen, P. A. J. Tilanus, and H. Zuidweg, "LOTOS extended with clocks," in *Formal Description Techniques, II*, pp. 179–194, IFIP, North-Holland, 1990.
11. T. Bolognesi, F. Lucidi, and S. Trigila, "From timed petri nets to timed LOTOS," in *Protocol Specification, Testing and Verification, X*, pp. 395–408, IFIP, North-Holland, 1990.
12. G. Leduc and L. Léonard, "A timed LOTOS supporting a dense time domain and including new timed operators," in *Formal Description Techniques, V*, pp. 87–102, IFIP, North-Holland, 1993.
13. P. C. Kanellakis and S. A. Smolka, "CCS expressions, finite state processes, and three problems of equivalence," *Information and Computation*, vol. 86, pp. 43–68, 1990.
14. N. Shiratori, H. Kaminaga, K. Takahashi, and S. Noguchi, "A verification method for LOTOS specifications and its application," in *Protocol Specification, Testing, and Verification, IX*, pp. 59–70, IFIP, North-Holland, 1990.
15. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
16. A. Nakata, T. Higashino, and K. Taniguchi, "LOTOS enhancement for specifying time-constraints among non-adjacent actions and verification of equivalence," IPSJ SIG Notes 93-DPS-61-19, Information Processing Society of Japan, July 1993.
17. J. F. Groote, "Transition system specifications with negative premises," in *Proc. of CONCUR '90*, vol. 458 of *Lecture Notes in Computer Science*, pp. 332–341, Springer-Verlag, 1990.
18. T. Higashino, J. Kitamiti, and K. Taniguchi, "Presburger arithmetic and its application to program developments," *Computer Software*, vol. 9, no. 6, pp. 31–39, 1992. (In Japanese).
19. K. Yasumoto, T. Higashino, T. Matsuura, and K. Taniguchi, "PROSPEX: A graphical LOTOS simulator for protocol specifications with N nodes," *IEICE Trans. on Communication*, vol. E75-B, no. 10, pp. 1015–1023, 1992.
20. T. Higashino, G. v. Bochmann, X. Li, K. Yasumoto, and K. Taniguchi, "Test system for a restricted class of LOTOS expressions with data parameters," in *Proc. 5th IFIP Int'l Workshop on Protocol Test Systems*, pp. 205–216, IFIP, North-Holland, Sept. 1992.
21. F. Moller and C. Tofts, "Relating processes with respect to speed," in *Proc. of CONCUR '91*, vol. 527 of *Lecture Notes in Computer Science*, pp. 424–438, Springer-Verlag, 1991.
22. S. Arun-Kumar and M. Hennessy, "An efficiency preorder for processes," *Acta Informatica*, vol. 29, pp. 737–760, 1992.