システムソフトウエアと
オペレーティング・システム 60－19
マルチメディア通信と分散処理 61－19
（１９９３．７．９）

隣接しない動作間の時間制約を記述するための LOTOS言語の拡張とその等価性の検証

中田明夫　東野輝夫　谷口健一
大阪大学基礎工学部情報工学科

本稿ではBasic LOTOSを拡張した言語LOTOS/Tを提案する．LOTOS/Tでは動作間の時間制約を１階述語論理式で記述できる．記述者は各動作が実行されるべき時間の論理的な関係のみを記述すれば良く，隣接しない動作間の時間制約を記述することができる．等号や不等号を制約式に用いることによって，インターバル，タイムアウトなどの時間性を容易に記述できる．まず，LOTOS/Tの構文と意味を形式的に与える．LOTOS/Tの意味はLTSを用いて定義し，LTSを動作式から構成するための推論規則を与える．そして，時間双模倣等価性と非時間双模倣等価性の２種類の等価性を導入し，LTSが有限ならばそれらの等価性が機械的証明可能であることを示す．実際的な記述例も与える．

LOTOS Enhancement for Specifying Time Constraints Among Non-adjacent Actions and Verification of Equivalence

Akio Nakata　Teruo Higashino　Kenichi Taniguchi

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

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. In LOTOS/T, the user only describes the logical relation of time at which each action must be executed. Time constraints among non-adjacent actions can be described. 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 semantics of LOTOS/T is defined using the Labelled Transition System (LTS). We give the inference rules for constructing the LTS's from given LOTOS/T expressions. We show that the LTS's can be constructed mechanically. Then we define both timed and untimed bisimulation equivalences. If the corresponding LTS's are finite, we can verify their equivalences mechanichally. A practical example is also given.

1 Introduction

Formal languages based on Process Algebra, such as CCS[10], CSP[6], ACP[2], LOTOS[8] 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 this paper, we propose a language 'LOTOS/T', which is a timed enhancement of Basic LOTOS. LOTOS/T allows us to describe time constraints by the 1st-order predicate logic formulas. We believe that the logic-based approach is very flexible and powerful, e.g. interval or primitive construct. The 1st-order predicate logic is well-studied, fits to automatic verification and is easy to describe complicated constraints in 'as is' way. Moreover, it will be easy to integrate logical time constraints into guard expressions of (Full-)LOTOS. 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 semantics of LOTOS/T is definied using 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 the predicates of 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 not directly followed by. For upward compatibility, if no predicate is 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.

Four equivalences are introduced, the first two are timed strong/weak bisimulation equivalence and the last two are untimed ones. Timed bisimulation equivalence is used for checking if two systems are equivalent and have the same time constraints. Untimed bisimulation equivalence is used for checking if two systems are equivalent in spite of the different time constraints. If the corresponding LTS's are finite, we can easily check the four bisimulation equivalences by the algorithms in [9, 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 and further problems are shown.

2 Definitions

2.1 Syntax

The syntax of new features we proposed to specify time constraints is defined as follows.

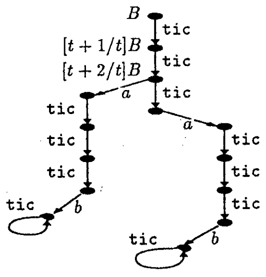

Figure 1: The semantics of B

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

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

where $a \in Act \cup \{i\}$ (Act denotes a finite set of all observable actions, i denotes an internal action) , $A \subset Act$, $k \in N$, 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[7]. □

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

Example 1 $B = a[2 \leq t \leq 3 \wedge x_0 = t]; b[t = x_0 + 3]; \textbf{stop}$

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 Fig. 1 denotes the operational semantics of B.

This LTS is obtained as follows. In Fig. 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, $a[t \leq 0 \wedge x_0 = t]; b[t+3 = x_0 + 3]; \textbf{stop}$ is obtained. If the tic was executed for $[t+3/t]B$, then the action a could not be executed. In this case, we say that the action a is urgent, and we assume that 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 to the variable t, representing the current time. Since $x_0 = t+3$, the value of the variable x_0 is fixed to 3, and $b[t+3 = 3+3]; \textbf{stop}$ is obtained as the new state (behaviour expression). So b is executed after 3 unit 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.

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.*

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

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

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

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 when a is executed. However, in the context C', the variable x_0 is defined because the value of x_0 has been fixed when 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.*

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

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

Hereafter, we define the set of predicates $P(t,\bar{x})$ which can be used in the predicate context C. In the following definition, $FVar(P)$ denotes a set of all free variables occurred in a predicate P.

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 $Pres(C)$.*

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.

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

1. *(decidability) For any $n \in N$ and $\bar{v}$, satisfiabilities of the two formulas $P(n,\bar{v},\bar{x}_u)$ and $(\mathcal{F}P)(n,\bar{v}) \stackrel{def}{=} \exists t' \exists \bar{x}_u [t' \geq n \wedge P(t',\bar{v},\bar{x}_u)]$ are decidable. Note that $\mathcal{F}$ denotes a mapping on formulas defined above.*
2. *(uniqueness of substitution) For any $n \in N$ 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: *Conditions 1 and 2 are used to make sure the semantical model of the expression is constructive.* $\Box$

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

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

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

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 [7]. Since it is known that satisfiability of Presburger Arithmetics is decidable [7], 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 [13].* $\Box$

$Pres(C)$ is useful for describing valid 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 "$a[t = x]; b[\bullet];$ **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 *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.*

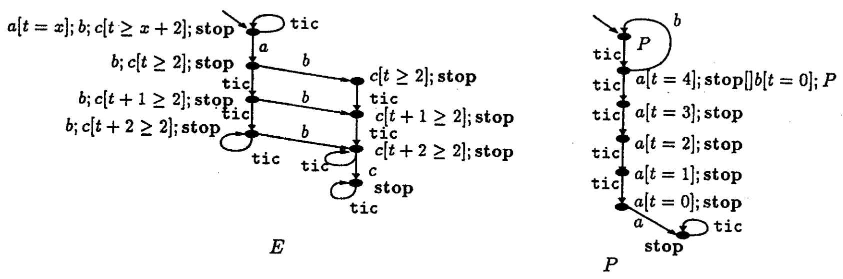

Figure 2: The semantics of E and P

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 Fig. 2.

In LOTOS/T, untimed or infinite behaviours may be described (for example, the processes E and P in Example 3).

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 Table 1).

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 valid, 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. when $\mathcal{F}P(1) \equiv \exists t' \exists \bar{x}[t' \geq 1 \wedge P(t', \bar{x})]$ 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[11]. For example, "$a[t = 1]$; **stop**" and "$b[t = 2]$; **stop**" are equivalent to "tic; a; **stop**" and "tic; tic; b; **stop**", respectively. However, "$a[t = 1]$; **stop**[]$b[t = 2]$; **stop**" is not equivalent to "tic; a; **stop**[]tic; tic; b; **stop**" because the choice is occurred at time 0. It must be equivalent to " tic; (a; **stop** [] tic; b; **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 $a; b[2 \leq t \leq 4]$; **stop**|[b]|$c; b[3 \leq t \leq 5]$; **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. The value passing form B_1 >> accept $\bar{x}$ in B_2 is also possible, although we omitted this for the simplicity. The definition of this form is analogous to Full-LOTOS.

2.2.8 Hide

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

2.2.9 Process Invocation

To preserve modular property of processes under global time, a process invocation acts exactly the same behaviour as the behaviour at time 0, no matter when it is invoked.

2.2.10 Example

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

Table 1: The inference rule of transition

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)
Parallel			
$\dfrac{B_1 \xrightarrow{\beta} B_1' \quad B_2 \xrightarrow{\beta} B_2'}{B_1\vert[A]\vert B_2 \xrightarrow{\beta} B_1'\vert[A]\vert B_2'}$ iff $\beta \in A \cup \{\delta\}$	(16)	$\dfrac{B_1 \xrightarrow{\mathtt{tic}} B_1' \quad B_2 \xrightarrow{\mathtt{tic}} B_2'}{B_1\vert[A]\vert B_2 \xrightarrow{\mathtt{tic}} B_1'\vert[A]\vert B_2'}$	(17)
$\dfrac{B_1 \xrightarrow{a} B_1'}{B_1\vert[A]\vert B_2 \xrightarrow{a} B_1'\vert[A]\vert B_2}$ iff $a \notin A \vee a = i$	(18)	$\dfrac{B_2 \xrightarrow{a} B_2'}{B_1\vert[A]\vert B_2 \xrightarrow{a} B_1\vert[A]\vert B_2'}$ iff $a \notin A \vee a = i$	(19)
$\dfrac{B_1\vert[\emptyset]\vert B_2 \xrightarrow{\alpha} B'}{B_1\vert\vert\vert B_2 \xrightarrow{\alpha} B'}$ iff $\alpha \in Act \cup \{\delta, \mathtt{tic}, i\}$	(20)	$\dfrac{B_1\vert[Act]\vert B_2 \xrightarrow{\alpha} B'}{B_1\vert\vert B_2 \xrightarrow{\alpha} B'}$ 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			
$\dfrac{[\bar{e}/\bar{x}]B\{g_1'/g_1, \cdots, g_k'/g_k\} \xrightarrow{\alpha} B'}{P[g_1', \cdots, g_k'](\bar{e}) \xrightarrow{\alpha} B'}$ iff $\alpha \in Act \cup \{\mathtt{tic}, \delta, i\} \wedge P[g_1, \cdots, g_k](\bar{x}) := B$ is a definition			(32)

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

and so on.

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

- $P \xrightarrow{\mathtt{tic}} a[t = 4]; \mathbf{stop}[]b[t = 0]; P$ (by rules (32), (13), (5)),
- $a[t = 4]; \mathbf{stop}[]b[t = 0]; P \xrightarrow{\mathtt{tic}} a[t = 3]; \mathbf{stop}$ (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 Fig 2, $a[t = x]; b; c[t \geq x + 2]; \mathbf{stop} \xrightarrow{\mathtt{tic}} a[t+1 = x]; b; c[t+1 \geq x+2]; \mathbf{stop}$ holds by the inference rules. Here, satisfiabilities of two predicates of the action a, $t = x$ and $t + 1 = x$, are equivalent, i.e.

$$\forall t[0 \leq t \Rightarrow \exists x[t = x] \equiv \exists x'[t + 1 = x']] \qquad (33)$$

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,

$$\forall t'[0 \leq t' \Rightarrow [t' \geq x + 2] \equiv [t' + 1 \geq x' + 2]] \qquad (34)$$

holds.

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

$$\begin{aligned}\forall t \forall t'[0 \leq t \wedge 0 \leq t' \Rightarrow \\ \exists x[t = x \wedge t' \geq x + 2] \equiv \\ \exists x'[t + 1 = x' \wedge t' + 1 \geq x' + 2]] \quad (35)\end{aligned}$$

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

Aging (replacing t with $t+1$) does not have effect on Process Invocation, because Process Invocation does not have the variable t literally until the process is invoked. For example, w.r.t. P in Fig 2, $P \xrightarrow{\mathtt{tic}} a[t = 4]; \mathbf{stop}[]b[t = 0]; P \xrightarrow{b} P$ holds by the inference rules. So the corresponding LTS has a cycle, as shown in Fig 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 $a \in Act \cup \{\delta, \mathtt{tic}\}$, 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 [13].

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

Example 4 *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 $\mathtt{tic}$ 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 $\mathtt{tic}$ is considered unobservable, and the other is *untimed weak bisimulation equivalence*, where both $\mathtt{tic}$ and i are considered unobservable.

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

$$B \xRightarrow{a} B' \overset{def}{\equiv} \begin{cases} B(\xrightarrow{\mathtt{tic}})^* \xrightarrow{a} (\xrightarrow{\mathtt{tic}})^* B', \\ \quad \text{if } a \in Act \cup \{\delta\} - \{\mathtt{tic}\} \\ B(\xrightarrow{\mathtt{tic}})^* B' \quad \text{if } a = \epsilon \ \square \end{cases}$$

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 $B_1 \xRightarrow{a} B_1'$, 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 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.:*

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

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 $B = b[t = 1]; \mathbf{stop}$.* □

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

Example 5 *Let B and D denote the following expressions, respectively:*

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

Then, B and D are untimed strong bisimulation equivalent because

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

is an untimed strong bisimulation which satisfies $\mathcal{BRD}$. □

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 four equivalences defined above are decidable.*

Proof. *Analogous to [9, 14].*

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 Fig. 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:

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

The second one is used for quitting, 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 Fig. 4.

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[5] 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[15] and a test system for LOTOS with data parameters[4]. 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.

In spite of its capability of higher level description, the semantic model of LOTOS/T is the same as LOTOS, because we consider discrete time. So many verification methods developed for LOTOS are still available for LOTOS/T.

We did not introduce timing-interaction operator defined in [3]. The strength of this is that locality of specification is preserved, as mentioned in [3] (but differs from Timed-Action LOTOS[3] 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[3] and Timed Petri Nets. But LOTOS/T is sufficient to use for formal description of specification and timed implementation to verify whether or not implementation satisfies untimed specification by checking untimed bisimulation equivalence.

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 [12, 1].

Future works are:

1. to provide timing of interactions (like `timer` operator of Timed-Interaction LOTOS) and non-deterministic execution delay (like Timed Petri-Net) to LOTOS/T for more practical timed description and performance evaluation, and
2. further exploration of untimed bisimulation equivalence.

References

[1] ARUN-KUMAR, S. AND HENNESSY, M. An efficiency preorder for processes, *Acta Informatica*, **29** (1992), 737–760.

[2] BERGSTRA, J. A. AND KLOP, J. W. Algebra of communicating processes with abstraction, *Theoretical Comput. Sci.*, **37** (1985), 77–121.

```
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

```
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

[3] Bolognesi, T., Lucidi, F. and Trigila, S. From Timed Petri Nets to Timed LOTOS, Protocol Specification, Testing and Verification, X (eds.Logrippo, L., Probert, R. L. and Ural, H.), IFIP, Elsevier Science Publishers B.V.(North-Holland) (1990).

[4] Higashino, T., Bochmann, G. v., Li, X., Yasumoto, K. and Taniguchi, K. Test System for a Restricted Class of LOTOS Expressions with Data Parameters, Proc. 5th IFIP Int'l Workshop on Protocol Test Systems, IFIP, North-Holland (Sept. 1992).

[5] Higashino, T., Kitamiti, J. and Taniguchi, K. Presburger Arithmetic and its Application to Program Developments, *Computer Software*, **9**, 6 (1992), 31–39, (In Japanese).

[6] Hoare, C. A. R. *Communicating Sequential Processes*, Prentice Hall (1985).

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

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

[9] Kanellakis, P. C. and Smolka, S. A. CCS Expressions, Finite State Processes, and Three Problems of Equivalence, *Information and Computation*, **86** (1990), 43–68.

[10] Milner, R. *A Calculus of Communicating Systems*, Vol. 92 of *Lecture Notes in Computer Science*, Springer-Verlag (1980).

[11] Moller, F. and Tofts, C. A Temporal Calculus of Communicating Systems, Proceedings CONCUR '90 (eds.Baeten, J. C. M. and Klop, J. W.), Vol. 458 of *Lecture Notes in Computer Science*, Springer-Verlag (1990).

[12] Moller, F. and Tofts, C. Relating Processes With Respect to Speed, Proceedings CONCUR '91 (eds.Baeten, J. C. M. and Groote, J. F.), Vol. 527 of *Lecture Notes in Computer Science*, Springer-Verlag (1991).

[13] Nakata, A., Higashino, T. and Taniguchi, K. Specification of Time-constrained concurrent system and Verification of Equivalence, Workshop on Fundamental Theory of Computer Science (LA symposium [Language and Automata symposium]), IEICE of JAPAN (July 1992), (In Japanese).

[14] Shiratori, N., Kaminaga, H., Takahashi, K. and Noguchi, S. A Verification Method for LOTOS Specifications and its application, Protocol Specification, Testing, and Verification, IX (eds.Brinksma, E., Scollo, G. and Vissers, C. A.), IFIP, Elsevier Science Publishers B.V.(North-Holland) (1990).

[15] Yasumoto, K., Higashino, T., Matsuura, T. and Taniguchi, K. PROSPEX: A Graphical LOTOS Simulator for Protocol Specifications with N nodes, *IEICE Trans. on Communication*, **E75-B**, 10 (1992), 1015–1023.