Control of Timed Discrete Event Systems with Ticked Linear Temporal Logic Constraints

Takuma Kinugawa∗, Kazumune Hashimoto∗, Toshimitsu Ushio∗

* Graduate school of Engineering Science, Osaka University, Japan.

Abstract: This paper presents a novel method of synthesizing a controller of a timed discrete event system (TDES), introducing a novel linear temporal logic (LTL), called ticked LTL. The ticked LTL is given as an extension to LTL, where the semantics is defined over a finite execution sequence. Differently from the standard LTL, the formula is defined as a variant of metric temporal logic formula, where the temporal properties are described by counting the number of tick in the execution sequence of the TDES. Moreover, we provide a scheme that encodes the problem into a suitable one that can be solved by an integer linear programming (ILP). The effectiveness of the proposed approach is illustrated through a numerical example of a path planning.

Keywords: Timed discrete event systems, linear temporal logic, integer linear programming

1. INTRODUCTION

A discrete event system (DES) is useful for the design of a logical high-level controller in many engineering fields such as manufacturing systems, traffic systems, and robotics (Cassandras and Lafortune (2008); Campos et al. (2014)). There are many formalisms of the DES, where its trajectories are represented by a sequence of states and/or events (Seatzu et al. (2013)). To model real-time systems, however, we also need information of times when state transitions occur. Many formalisms including the temporal information in the models of the DES have been proposed (Bakker et al. (1991)). Alur and Dill (1994) proposed a timed automaton that is an extension of an automaton by introducing real-valued variables indicating times elapsed since events occur. The timed automaton is a dense time model and, as an abstraction of the dense time, a fictitious clock has been introduced (Raskin and Schobbens (1997); Henzinger et al. (1992)). Ostrouf and Wonham (1990) introduced a timed transition model (TTM) where a discrete time elapsed is described by a special event tick. Moreover, Brandin and Wonham (1994) formulated timed discrete event systems (TDES) by a timed transition graph that is a transition graph with state transitions by the event tick.

On the other hand, in computer science, the temporal logic (TL) has been developed to specify the trajectories of systems that we verify (Baier and Katoen (2008); Clarke, Jr. et al. (2018)). For example, in model checking of a non-terminating program, the specification is described by a TL formula and the correctness of the program is verified. So, the satisfaction relation for the TL formula is defined over infinite trajectories of the verified system. Many different temporal logics have been proposed and their expressiveness have been studied. Among them, the linear temporal logic (LTL) is often used because it can describe many properties that specifications often requires such as safety, stability, and progress. Many approaches to LTL model checking where the specification is described by an LTL formula have been proposed. A basic idea to solve the LTL model checking is a usage of a tableau and an automata-theoretic approach is widely used. As alternative approaches, symbolic model checking using binary decision diagrams and bounded model checking using a SAT solver have been developed. In the bounded model checking, we search a lasso type trajectory that is a counterexample of the LTL specification. Biere et al. (2006) proposed efficient encodings for the bounded LTL model checking.

The TL formula has been also leveraged as formal description of a control specification in the DES (Thistle and Wonham (1986); Jiang and Kumar (2006); Sakakibara and Ushio (2018)). Recently, the formal synthesis of control systems has been much attention to (Belta et al. (2017)). For example, Kress-Gazit et al. (2009) describes a high-level specification by an LTL formula and constructed a hybrid controller satisfying the specification. Wongpiromsam et al. (2012) proposed receding horizon control for an LTL control specification. There is a method to generate trajectories of continuous systems with encoding specifications formulated by LTL formulas as mixed integer linear programming constraints (Wolff et al. (2014)). Many path planning problems of mobile robots can be restricted to a finite horizon. A controller synthesis problem where a control specification is described by a TL formula, called an LTL formula, not for infinite trajectories but for finite ones has been proposed (Zhu et al. (2017)). Li et al. (2019) presented SAT-based LTL model checking.

In verification and control of real-time systems, however, control specifications depend not only on logical constraints but also on the timing at which each event occurs. Koymans (1990) proposed metric TL (MTL) for

Copyright lies with the authors
a timed state sequence with a function that assigns the time stamp to each state. Maler and Nickovic (2004) introduced a signal TL that specifies dense-time real-valued signals. Raskin and Schobbens (1997) considered the case where the real-time information is described based on a fictitious clock. Ostooff (1990) defined real-time TL for real-time system modeled by the TTM. Barbeau et al. (1998) dealt with a synthesis problem of controllers for TDES with a control specification described by an MTL formula. Dhananjayan and Seow (2014) proposed an MTL specification interface that translates an MTL specification to a finite timed transition graph used in the synthesis of a timed supervisor.

In this paper, we provide a novel approach to controller synthesis for TDES, introducing a novel LTL called ticked LTL. As with the standard LTL (Zhu et al. (2017)), the formula will be interpreted over the finite execution sequence, which, as previously mentioned, may be a natural assumption in many path planning problems. In contrast to the standard LTL, the formula in this paper is given as an MTL, where temporal properties are described by counting the number of the event tick in the sequence of the TDES. As we will see later, the problem is formulated to find a suitable (finite) execution sequence of the TDES, such that a given ticked LTL formula is satisfied. Moreover, we provide an encoding scheme such that the problem can be translated into an integer linear programming (ILP). Finally, the effectiveness of the proposed approach is illustrated through a numerical example of a path planning.

The rest of this paper is organized as follows. In Section 2, we introduce TDES formulated by Brandin and Wonham (1994). In Section 3, we define syntax and semantics of the ticked LTL. In Section 4, we provide the problem and an encoding scheme so that it can be translated to the integer linear programming. In Section 5, we apply the proposed approach to a path planning problem of an agent. Section 6 concludes the paper.

2. TIMED DISCRETE EVENT SYSTEM

In this section, we review basic definitions of untimed and timed discrete event systems.

2.1 Discrete event systems

Let us first define an untimed discrete event system (DES) by a transition system:

Definition 1. (Untimed DES). The untimed DES is a tuple $G_{act} = (S_{act}, \Sigma_{act}, \delta_{act}, s_0, L_{act}, A_{act})$, where $S_{act}$ is a set of states, $\Sigma_{act}$ is a set of events, $\delta_{act} : S_{act} \times \Sigma_{act} \rightarrow S_{act}$ is a transition function, $s_0 \in S_{act}$ is the initial state, $L_{act}$ is the labeling function, $A_{act}$ is a set of atomic propositions and $T_{act} : S_{act} \rightarrow 2^{AP_{act}}$ is a labeling function.

Next, we incorporate some timing properties in $G_{act}$. To this end, assume that each event $\sigma \in \Sigma_{act}$ is enabled during a specified time interval $[l_{\sigma}, u_{\sigma}]$, where $l_{\sigma}, u_{\sigma} \in \mathbb{N}$ with $l_{\sigma} \leq u_{\sigma}$ are called the lower time and the upper time bound, respectively. In particular, the event $\sigma$ is called a prospective (resp. remote) event if $u_{\sigma} \in \mathbb{N}$ (resp. $u_{\sigma} = \infty$). Let $\Sigma_{spe}$ and $\Sigma_{rem} \subseteq \Sigma_{act}$ be the sets of prospective and remote events, respectively. Note that $\Sigma_{spe} \cup \Sigma_{rem} = \Sigma_{act}$. Then, we introduce the following timer interval $T_{\sigma}$ for each event $\sigma \in \Sigma_{act}$:

$$T_{\sigma} = \begin{cases} [0, u_{\sigma}] & \text{if } \sigma \in \Sigma_{spe}, \\ [0, l_{\sigma}] & \text{if } \sigma \in \Sigma_{rem}. \end{cases}$$

Moreover, we introduce a special event tick, which represents the global clock and will be utilized as an additional event to $\Sigma_{act}$. Based on the above definitions, a timed DES corresponding to $G_{act}$ is defined as follows:

Definition 2. (Timed DES). A timed DES (TDES) corresponding to $G_{act}$ is a tuple $G = (S, \Sigma, \delta, s_0, AP, L)$ where $S = S_{act} \times \prod_{\sigma \in \Sigma_{act}} T_{\sigma}$ is a set of states, $\delta : S \times \Sigma \rightarrow S$ is a transition function, $s_0 \in S$ is the initial state, where $s_0 = (s_{0,act}, \{t_{\sigma,0} \mid \sigma \in \Sigma_{act}\})$, and $t_{\sigma,0}$ is given by $u_{\sigma}$ if $\sigma \in \Sigma_{spe}$ or $l_{\sigma}$ if $\sigma \in \Sigma_{rem}$. $AP = AP_{act}$ is a set of atomic propositions and $L : S \rightarrow 2^{AP}$ is a labeling function, where $L(s) = L_{act}(a)$ and $s = (a, \{t_{\sigma} \mid \sigma \in \Sigma_{act}\}) \in S$.

The concrete definition of the transition function $\delta$ is omitted in this paper and the reader is referred to Brandin and Wonham (1994) for details.

Example 1. Fig. 1 shows an example of a DES and the corresponding TDES. The set of states $S_{act}$ and the set of events $\Sigma_{act}$ are given by $S_{act} = \{s\}$ and $\Sigma_{act} = \{\alpha, \beta\}$. The lower time bound and the upper time bound for $\alpha$ and $\beta$ are given by $l_{\alpha} = 1, u_{\alpha} = \infty, l_{\beta} = 1, u_{\beta} = 2$. For example, the remote event $\alpha$ with $l_{\alpha} = 1, u_{\alpha} = \infty$ implies that the event $\alpha$ can occur at any time after the event tick occurs once. In Fig. 1(b), a pair $(s, [t_{\alpha}, t_{\beta}])$ represents a state of $G$ with $s \in S_{act}$ and $[t_{\alpha}, t_{\beta}] \in T_{\alpha} \times T_{\beta}$.

A finite execution $\pi$ of $G$ is a finite sequence of alternating states and events

$$\pi = s(0), e(1), s(1), \ldots, e(H), s(H),$$

where $H \in \mathbb{N}_{>0}$, $s(k) \in S$, $\forall k \in \{0, \ldots, H\}$, $e(k) \in \Sigma$, $\forall k \in \{1, \ldots, H\}$ and $s(0) = s_0$, $(s(k-1), e(k), s(k)) \in \delta$, $\forall k \in \{1, \ldots, H\}$. Here, $H$ is called the length or horizon of $\pi$. Moreover, the corresponding sequence of states

$$s(0), s(1), \ldots, s(H)$$

Fig. 1. An example of DES and TDES (for brevity, the atomic propositions are omitted and are not shown in the figure).
is called a trajectory of $G$. For given (3) and $k \in \{0, \ldots, H\}$, let $\pi(k) = s(k)$, and

$$\pi(k \ldots) = s(k), e(k+1), s(k+1), \ldots, e(j), s(j).$$

Moreover, for given (3) and $k, j \in \{0, \ldots, H\}$ with $k \leq j$, let $\pi(k \ldots j)$ be the partial suffix given by

$$\pi(k \ldots j) = s(k), e(k+1), s(k+1), \ldots, e(j), s(j).$$

For example, if $\pi = a, tick, a, \sigma, b, tick, a$ with $AP = \{a, b\}$ and $\Sigma = \{\sigma\} \cup \{tick\}$, we have $count_{\pi}(0,3) = 2$, $count_{\pi}(1,3) = 1$ since $\pi(0 \ldots 3) = a, tick, a, \sigma, b, tick, a$ and $\pi(1 \ldots 3) = a, \sigma, b, tick, a$. Note that we have $count_{\pi}(k, k) = 0$, $\forall k \in \{0, \ldots, H\}$, since $\pi(k \ldots k) = s(k)$ and so no events occur in $\pi(k \ldots k)$.

### 3. TICKED LINEAR TEMPORAL LOGIC

We now introduce a novel temporal logic called ticked LTL$_f$ over TDES. As will be seen below, this formula is interpreted over a finite execution (3), and provides an extension of the LTL$_f$ formula Zhu et al. (2017), in the sense that we incorporate some timing properties via the event tick. First, we define its syntax as follows.

**Definition 3.** (Syntax of ticked LTL$_f$). A ticked LTL$_f$ formula over sequences of atomic propositions is recursively defined according to the following grammar:

$$\phi ::= True \mid ap \mid \neg \phi \mid \phi_1 \land \phi_2 \mid \phi_1 U[m,n] \phi_2,$$

where $ap \in AP$, $m$ and $n$ are nonnegative integers with $m \leq n$.

Note that, while we do not include the operator $\bigcirc$ (next) in the syntax, $F_{[1,1]}$ can be used as the operator having the same meaning as $\bigcirc$. Additional boolean operators are: $\phi_1 \lor \phi_2 ::= \neg (\neg \phi_1 \land \neg \phi_2)$, $\phi_1 \rightarrow \phi_2 ::= \neg \phi_1 \lor \phi_2$, $\phi_1 \leftrightarrow \phi_2 ::= (\phi_1 \rightarrow \phi_2) \land (\phi_2 \rightarrow \phi_1)$. Moreover, other temporal operators, such as $\bigcirc_{[m,n]}$ (future) and $\sqcap_{[m,n]}$ (globally) are defined by

$$\bigcirc_{[m,n]} \phi ::= True U[m,n] \phi, \quad \sqcap_{[m,n]} \phi ::= \neg \bigcirc_{[m,n]} \neg \phi.$$

Its semantics is defined over a finite execution in (3) and is formally given as follows.

**Definition 4.** (Semantics of ticked LTL$_f$). Given a finite execution $\pi = s(0), e(1), s(1), \ldots, e(H), s(H)$, the satisfaction of the ticked LTL$_f$ formula $\phi$ for the $k$-th suffix of $\pi$ ($0 \leq k \leq H$), denoted as $\pi(k \ldots) \models \phi$, is defined recursively as follows:

- $\pi(k \ldots) \models True$,
- $\pi(k \ldots) \models ap$ if and only if ap $\in L(\pi(k))$,
- $\pi(k \ldots) \models \neg \phi$ if and only if $\pi(k \ldots) \not\models \phi$,
- $\pi(k \ldots) \models \phi_1 \land \phi_2$ if and only if $\pi(k \ldots) \models \phi_1$ and $\pi(k \ldots) \models \phi_2$,
- $\pi(k \ldots) \models \phi_1 U[m,n] \phi_2$ if and only if there exist $j \in [k, H]$ such that $m \leq count_{\pi}(k, j) \leq n$, $\pi(j \ldots) \models \phi_2$ and $\pi(i \ldots) \models \phi_1$, $\forall i \in [k, j - 1]$.

Intuitively, the formula $\phi_1 U[m,n] \phi_2$ indicates that, $\phi_1$ holds true until $\phi_2$ holds true during the interval that the number of ticked events is between $m$ and $n$. We denote by $\pi \models \phi$ if and only if $\pi(0 \ldots) \models \phi$.

**Example 2.** Consider a finite execution sequence: $\pi = a, tick, a, \sigma, b, tick, a$. Also, consider a ticked LTL$_f$ formula $\phi = ap_1 U ap_2$ with $AP = \{ap_1, ap_2\}$, $S = \{a, b\}$, $\Sigma = \{\sigma\} \cup \{tick\}$ and $L(a) = \{ap_1\}$, $L(b) = \{ap_2\}$. Then, $\pi(0 \ldots) \models \phi$ if and only if $\pi(k \ldots) \models \phi$ and so no events occur in $\pi(k \ldots k)$.

**4. CONTROLLER SYNTHESIS UNDER LTL$_F$ CONSTRAINTS**

Using the ticked LTL$_f$ introduced in the previous section, we consider the following problem.

**Problem 1.** Given a TDES $G$, a ticked LTL$_f$ formula $\phi$ and a horizon $H > 0$, synthesize a finite execution $\pi$ of $G$ with the horizon $H$ such that $\pi \models \phi$.

To solve Problem 1, we translate a finite trajectory of the TDES $G$, the counting function $count_{\pi}$, and the ticked LTL$_f$ formula $\phi$ into a set of inequality and equality constraints that can be solved by an integer linear programming (ILP) problem. Note that an ILP problem can be solve by a SAT-solver because a SAT-solver gets remarkable developments recently. Details for the encodings are described below.

**4.1 Encoding the trajectory of $G$**

To encode the trajectory of $G$, we denote by $A \in \{0, 1\}^{N \times N}$ with $N = |S|$ the adjacency matrix of the graph in accordance with $G$, i.e., letting $S = \{s_1, \ldots, s_n\}$, we have $A_{ij} = 1$ (the $(i, j)$-component of $A$ is 1) if and only if there exists $\sigma \in \Sigma$ such that $s_i \in \delta(s_j, \sigma)$ and 0 otherwise. Moreover, we introduce $H + 1$ binary vectors $w(k) \in \{0, 1\}^N$, $k \in \{0, \ldots, H\}$ to represent the state of $G$ at $k$, where, for each $k \in \{0, \ldots, H\}$, the vector $w(k)$ includes only one non-zero component. That is, if $\pi$ is given by (3), we have $w_i(k) = 1$ (the $i$-th component of $w(k)$) if and only if $\pi(k) = s_i$, and 0 otherwise. The trajectory of the states can be then encoded as follows.

$$w(k + 1) \leq A^T w(k), \quad 1^T w(k) = 1,$$

where $1^T$ is the $N$-dimensional vector whose all elements are 1.

**4.2 Encoding the counting function**

Let $c(k,j) \in \mathbb{N}$ for $k, j \in \{0, \ldots, H\}$ with $k \leq j$ be integer variables that represent the number of tick events in the trajectory $\pi(k \ldots j)$, i.e, $c(k, j) = m$ if and only if $count_{\pi}(k, j) = m$. The $c(k, j)$ represented as an ILP problem constraint as follows. First, we introduce $H$ binary variables $z_{c}(k) \in \{0, 1\}$, for $k \in \{1, \ldots, H\}$ in order to represent the occurrence of tick in the sequence of events, i.e., if $\pi$ is given by (3), we have $z_{c}(k) = 1$ if and only if $c(k) = tick$. Using $z_{c}(k), k \in \{0, \ldots, H\}$, $c(k, j)$ is then given by

$$c(k, j) = \sum_{i=k+1}^{j} z_{c}(i)$$
for \( k, j \in \{0, \ldots, H \} \) with \( k < j \), and \( c(k, k) = 0 \), \( \forall k \in \{0, \ldots, H \} \). The variables \( z_e(k), k \in \{1, \ldots, H \} \) encode whether \( e(k) \) is the event \( \text{tick} \) or not. First, let \( \alpha \in \{0, 1\}^N \) be a binary vector, such that \( \alpha_0 = 1 \) (the \( i \)-th component of \( \alpha \) is 1) if and only if \( \delta(s_i, \text{tick}) \) (i.e., the event \( \text{tick} \) occurs at \( s_i \)). Moreover, let \( \beta \in \{0, 1\}^N \) be a binary vector such that \( \beta_1 = 1 \) (the \( i \)-th element of \( \beta \) is 1) if and only if there exists \( s_j \in S \) such that \( s_i = \delta(s_j, \text{tick}) \) (i.e., there exists a state that can transition to \( s_j \) through the event \( \text{tick} \)).

Then, \( z_e(k) = 1 \) if and only if
\[
\alpha^T w(k-1) = 1 \land \beta^T w(k) = 1.
\]
Thus, \( z_e(k) = 1 \) if and only if the following equations hold.
\[
\begin{align*}
z_e(k) &\leq \alpha^T w(k-1), \quad (9) \\
z_e(k) &\leq \beta^T w(k), \quad (10) \\
z_e(k) &\geq -1 + \alpha^T w(k-1) + \beta^T w(k). \quad (11)
\end{align*}
\]

4.3 Encoding the ticked LTL\(_f\) formula

We introduce \( H + 1 \) binary variables \( z_\phi(k) \in \{0, 1\} \) for \( k \in \{0, 1, \ldots, H \} \), such that \( z_\phi(k) = 1 \) if and only if \( \pi(k) \) satisfies \( \phi \). The encodings for the ticked LTL\(_f\) formula \( \phi \) can be recursively given as follows:

(atomic proposition): Let \( \phi = ap \in AP \) and \( v \in \{0, 1\}^N \) be a binary vector such that \( v_i = 1 \) (the \( i \)-th component of \( v \) is 1) if and only if \( ap \in L(s_i) \). Then, the satisfaction of the formula \( \phi \) is encoded as follows:
\[
\begin{align*}
v^T w(k) &\geq z_\phi(k), \quad (12) \\
v^T w(k) &< z_\phi(k) + 1. \quad (13)
\end{align*}
\]

(negation): Let \( \phi = \neg \psi \). Then, the satisfaction of \( \phi \) is encoded as
\[
z_\phi(k) = 1 - z_\psi(k). \quad (14)
\]

(conjunction): Let \( \phi = \bigwedge_{i=1}^L \psi_i \). Then,
\[
z_\phi(k) \leq z_\psi_i(k), \quad \forall i \in \{1, \ldots, L\},
\]
\[
z_\phi(k) \geq 1 - L + \sum_{i=1}^L z_\psi_i(k). \quad (15)
\]

(disjunction): Let \( \phi = \bigvee_{i=1}^L \psi_i \). Then,
\[
z_\phi(k) \geq z_\psi_i(k), \quad \forall i \in \{1, \ldots, L\},
\]
\[
z_\phi(k) \leq \sum_{i=1}^L z_\psi_i(k). \quad (16)
\]

With a slight abuse of notation, Boolean operators are used for binary variables. For example, when we consider \( \phi = \bigwedge_{i=1}^L \psi_i \), we write \( z_\phi = \bigwedge_{i=1}^L z_\psi_i \) instead of (15).

4.4 Overall problem

Based on the above encodings, we formulate the following ILP problem to find a finite execution whose horizon is \( H \) satisfying the ticked LTL\(_f\) formula \( \phi \).

Find:
\[
w(k), z_\phi(k), k \in \{0, \ldots, H\}, \quad (20)
\]
\[
z_e(k), k \in \{1, \ldots, H\}, \quad (21)
\]
subject to the following constraints:
\[
(7) - (8), \quad ILP(\phi), \quad z_\phi(0) = 1, \quad (23)
\]
where \( ILP(\phi) \) is the constraints of an ILP problem for ticked LTL\(_f\) formula \( \phi \) generated from the procedure described in Section 4.3. The above problem can be solved by several off-the-shelf tools, such as Gurobi (available: https://www.gurobi.com), z3 Moura and Björner (2008), and so on.

5. APPLICATION TO PATH PLANNING

In this section, we demonstrate the effectiveness of the proposed approach through a numerical simulation of a path planning problem. The simulation was run by a machine with Intel Core i5 8400 and two 8GB memories.

5.1 Setting of TDES

Behavior of an agent (e.g., robot, drone, etc.) is first modeled by an untimed transition system \( G_{act} \), as shown in Fig 2. In the figure, each node represents the state of the agent, and each edge represents the transition between them. More specifically, if the state of the agent is \( p_i \) (\( i \in \{1, \ldots, 4\} \)), it means that the agent is in the location \( p_i \). Moreover, if the state is \( p_{ij} \), it means that the agent is on the way from \( p_i \) to \( p_j \). The symbols \( \text{move}_{ij} \) and \( \text{reach}_{ij}, i, j \in \{1, \ldots, 4\} \) represent the events that are
associated with the edges. More specifically, the event \textit{move}_{ij} indicates that the agent decides to move from \(p_i\) to \(p_j\), and the event \textit{reach}_{ij} indicates that the agent reaches \(p_j\). The set of atomic propositions is given by \(AP_{act} = \{ap_1, ap_2, ap_3, ap_4\}\), and the labeling function is \(L_{act}(p_i) = \{ap_i\}, \forall i \in \{1,\ldots, 4\}\). The initial state is \(p_1\).

The timer interval \(T_{\tau}\) is then defined as follows: if \textit{reach}_{ij} for \(i,j \in \{1,\ldots, 4\}\) is defined in Fig. 2, \(T_{\text{reach}_{ij}}\) is given by
\[
T_{\text{reach}_{12}} = [2, \infty], T_{\text{reach}_{14}} = [1, \infty], T_{\text{reach}_{21}} = [3, \infty], \\
T_{\text{reach}_{23}} = [2, \infty], T_{\text{reach}_{32}} = [3, \infty], T_{\text{reach}_{34}} = [2, \infty], \\
T_{\text{reach}_{41}} = [2, \infty], T_{\text{reach}_{43}} = [1, \infty].
\] (24)

For example, \(T_{\text{reach}_{12}} = [2, \infty]\) implies that, if the state of the agent is \(p_{12}\) (i.e., it is on the way from \(p_1\) to \(p_2\)), the event \textit{reach}_{12} can occur at any time after \(2\) ticks. In other words, the agent takes at least \(2\) ticks to move from \(p_1\) to \(p_2\). On the other hand, if \textit{move}_{ij} for \(i,j \in \{1,\ldots, 4\}\) is defined in Fig. 2, \(T_{\text{move}_{ij}}\) is then given by
\[T_{\text{move}_{ij}} = [0, \infty].\] (25)

(25) indicates that, if the state of the agent is \(s_1\), the event \textit{move}_{ij} can occur at any time (i.e., after any number of the event \textit{tick} occurs). Based on the above definitions, the corresponding timed transition system \(G\) is constructed according to Definition 2.

5.2 Simulation results

We first consider the following specification: \(\phi_1 = \Diamond_{[1.5]} ap_2 \land \Diamond_{[1.5]} ap_4\). That is, starting from the initial position (i.e., \(p_1\)), the agent must reach \(p_2\) and \(p_4\) while the number of the event \textit{tick} is between 1 and 5. The corresponding ILP problem is solved with different selections of \(H\), in order to find the execution satisfying \(\phi_1\). Specifically, starting from \(H = 5\), we solve the corresponding ILP problem and we increment the horizon until the execution satisfying \(\phi_1\) has been found. The execution was found with \(H = 11\) and is illustrated in Fig. 3. The figure shows that \(\Diamond_{[1.5]} ap_4\) is satisfied with the total number of \textit{tick} given by 1 (right figure of Fig. 3(a)), and \(\Diamond_{[1.5]} ap_2\) is satisfied with the total number of \textit{tick} given by 5 (right figure of Fig. 3(b)). The result of execution is concretely given by
\[
\pi_1 = p_1, move_{14}, p_{14}, tick, p_{14}, reach_{14}, p_{14}, ...
move_{41}, p_4, tick, p_4, tick, p_4, reach_{41}, p_1, ...
move_{12}, p_{12}, tick, p_{12}, tick, p_{12}, reach_{12}, p_2. \tag{26}
\]

Therefore, the result of execution is shown to satisfy \(\phi_1\). (26) implies that the agent aims to satisfy \(ap_4\) and then satisfy \(ap_2\). Alternatively, the agent might instead aim to satisfy \(ap_2\) and then \(ap_4\). However, from (24), this would then require at least \(2 + 3 + 1 = 6\) ticks to reach \(p_4\), which means that the formula \(\Diamond_{[1.5]} ap_4\) does not hold. That is, if the execution were generated such that the agent aims to satisfy \(ap_2\) and then \(ap_4\), it would then violate \(\phi_1\). Hence, it is shown that the ILP problem could appropriately select the execution such that the agent could satisfy the given specifications.

As another example, we consider \(\phi_2 = (\neg ap_2) U_{[3.5]} ap_3\), which indicates that the agent must avoid \(p_2\) until the agent reaches \(p_3\) with the number of \textit{tick} being from 3 to 5.

The execution satisfying \(\phi_2\) is found with \(H = 10\) and the result is shown in Fig. 4. The figure shows that the agent reaches \(p_3\) while avoiding \(p_2\) with total number of \textit{tick} given by 3 (Fig. 4(b)). The result of execution is concretely given by
\[
\pi_2 = p_1, tick, p_1, move_{14}, p_{14}, tick, p_{14}, reach_{14}, p_{14}, ...
move_{43}, p_{43}, tick, p_{43}, reach_{43}, p_3, move_{32}, p_{32}, ...
tick, p_{32}. \tag{27}
\]

Therefore, it is shown that the agent satisfies the formula \(\phi_2\).
(a) Partial execution of $\pi_2$ until the number of the event tick is 1.

(b) Partial execution of $\pi_2$ until the number of the event tick is 3.

Fig. 4. Result of execution $\pi_2$ by solving the ILP problem.

In the figure, red nodes and edges represent the path that the agent traverses according to $\pi_2$.

The authors are supported by ERATO HASUO Meta-mathematics for Systems Design Project (No. JPMJER1603), JST.

REFERENCES


