Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

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_f. The ticked LTL_f is given as an extension to LTL_f, where the semantics is defined over a *finite* execution sequence. Differently from the standard LTL_f, 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)). Ostroff and Wonham (1990) introduced a timed transition model(TTM) where a discrete time elapse 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_f formula, not for infinite trajectories but for finite ones has been proposed (Zhu et al. (2017)). Li et al. (2019) presented SAT-based LTL_f 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 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. Ostroff (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_f. As with the standard LTL_f(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_f, 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_f 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_f. In Section 4, we provide the problem and an encoding scheme so that it can be translated in 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

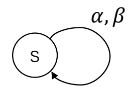
In this section, we review basic definitions of untimed and timed discrete event systems.

2.1 Discrete event systems

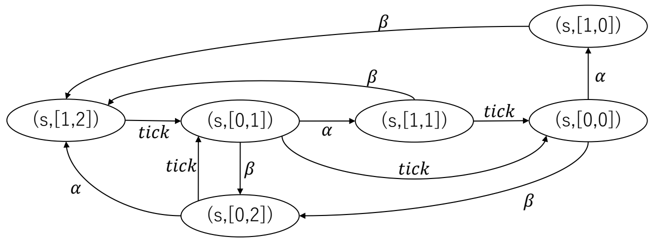
Let us first define an untimed discrete event system (DES) by a transition system:

Definition 1. (Untimed DES). The untimed DES is a tuple $G_{act} = (S_{act}, \Sigma_{act}, \delta_{act}, s_{0,act}, L_{act}, A_{act})$, where S_{act} is a set of states, Σ_{act} is a set of events, $\delta_{act} : S_{act} \times \Sigma_{act} \to S_{act}$ is a transition function, $s_{0,act}$ is the initial state, AP_{act} is a set of atomic propositions and $L_{act} : S_{act} \to 2^{AP_{act}}$ 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 $l_\sigma \in \mathbb{N}$, $u_\sigma \in \mathbb{N} \cup \{\infty\}$ with $l_\sigma \leq u_\sigma$ are called the lower time and the upper time bound, respectively. In particular, the event σ is called a prospective (*r*esp. remote) event if $u_\sigma \in \mathbb{N}$ (*r*esp. $u_\sigma = \infty$). Let Σ_{spe} and $\Sigma_{rem} \subseteq \Sigma_{act}$ be the sets



(a) An untimed DES G_{act}.



(b) The TDES G corresponding to G_{act} in Fig. 1(a).

Fig. 1. An example of DES and TDES (for brevity, the atomic propositions are omitted and are not shown in the figure).

of prospective and remote events, respectively. Note that $\Sigma_{spe} \cup \Sigma_{rem} = \Sigma_{act}$. Then, we introduce the following timer interval T_σ for each event $\sigma \in \Sigma_{act}$:

$$T_\sigma = \begin{cases} [0,\ u_\sigma] \text{ if } \sigma \in \Sigma_{spe}, \\ [0,\ l_\sigma] \ \text{ if } \sigma \in \Sigma_{rem}. \end{cases} \tag{1}$$

Moreover, we introduce a special event *tick*, which represents the global clock and will be utilized as an additional event to Σ_{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, $\delta : S \times \Sigma \to S$ is a transition function, $s_0 \in S$ is the initial state, where $s_0 = (s_{0,act}, \{t_{\sigma,0} | \sigma \in \Sigma_{act}\})$, and $t_{\sigma,0}$ is given by u_σ if $\sigma \in \Sigma_{spe}$ or l_σ if $\sigma \in \Sigma_{rem}$, $AP = AP_{act}$ is a set of atomic propositions and $L : S \to 2^{AP}$ is a labeling function, where $L(s) = L_{act}(a)$ and $s = (a, \{t_\sigma | \sigma \in \Sigma_{act}\}) \in S$.

The concrete definition of the transition function δ 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 Σ_{act} are given by $S_{act} = \{s\}$ and $\Sigma_{act} = \{\alpha, \beta\}$. The lower time bound and the upper time bound for α and β are given by

$$l_\alpha = 1, u_\alpha = \infty, l_\beta = 1, u_\beta = 2. \tag{2}$$

For example, the remote event α with $l_\alpha = 1, u_\alpha = \infty$ implies that the event α 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* π *of* G is a finite sequence of alternating states and events

$$\pi = s(0), e(1), s(1), \ldots, e(H), s(H), \tag{3}$$

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

$$s(0), s(1), \ldots, s(H) \tag{4}$$

is called a *trajectory* of G. For given (3) and $k \in \{0, ..., H\}$, let $\pi(k) = s(k)$, and

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

i.e., $\pi(k...)$ denotes the k-th suffix of π. Moreover, for given $k, j \in \{0, ..., H\}$ with $k \leq j$, let $\pi(k...j)$ be the partial suffix given by

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

Moreover, for given (3) and $k, j \in \{0, ..., H\}$ with $k \leq j \leq H$, let $count_\pi(k, j)$ denote the number of the event *tick* included in $\pi(k...j)$.

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

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:

$$\phi := True \mid ap \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[m,n]} \phi_2, \tag{5}$$

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: $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\phi_1 \rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$, $\phi_1 \leftrightarrow \phi_2 := (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$. Moreover, other temporal operators, such as $\diamond_{[m,n]}$ (future) and $\square_{[m,n]}$ (globally) are defined by

$$\diamond_{[m,n]}\phi := True U_{[m,n]} \phi, \ \square_{[m,n]}\phi := \neg\diamond_{[m,n]}\neg\phi. \tag{6}$$

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 ϕ for the k-th suffix of π ($0 \leq k \leq H$), denoted as $\pi(k...) \models \phi$, is defined recursively as follows:

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

Intuitively, the formula $\phi_1 U_{[m,n]} \phi_2$ indicates that, ϕ_1 holds true until ϕ_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 $\pi(0...) \models \phi$.

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, $\pi(0...) (= a, tick, a, \sigma, b, tick, a)$ satisfies ϕ, since a holds true until b holds true while the number of *tick* counted from $\pi(0)$ is 1, i.e., $count_\pi(0, 2) = 1 \in [1, 3]$. However, $\pi(1...) (= a, \sigma, b, tick, a)$ does *not* satisfy ϕ, since b holds true while the number of *tick* counted from $\pi(1)$ is 0, i.e., $count_\pi(1, 2) = 0 \notin [1, 3]$. □

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 ϕ and a horizon $H > 0$, synthesize a finite execution π 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 ϕ 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 $S = \{s_1, ..., s_N\}$, we have $A_{i,j} = 1$ (the (i,j)-component of A is 1) if and only if there exists $\sigma \in \Sigma$ such that $s_j \in \delta(s_i, \sigma)$, and 0 otherwise. Moreover, we introduce $H+1$ binary vectors $w(k) \in \{0,1\}^N$, $k \in \{0, ..., H\}$ to represent the state of G at k, where, for each $k \in \{0, ..., H\}$, the vector $w(k)$ includes only one non-zero component. That is, if π is given by (3), we have $w_i(k) = 1$ (the i-th component of $w(k)$ is 1) if and only if $s(k) = s_i$, and 0 otherwise. The trajectory of the states can be then encoded as follows.

$$w(k+1) \leq A^\mathsf{T} w(k), \ 1_N^\mathsf{T} w(k) = 1, \tag{7}$$

where 1_N is the N-dimensional vector whose all elements are 1.

4.2 Encoding the counting function

Let $c(k,j) \in \mathbb{N}$ for $k, j \in \{0, ...H\}$ with $k \leq j$ be integer variables that represent the number of *tick* events in the trajectory $\pi(k...j)$, 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 $z_e(k) \in \{0,1\}$, for $k \in \{1, ..., H\}$ in order to represent the occurrence of *tick* in the sequence of events, i.e., if π is given by (3), we have $z_e(k) = 1$ if and only if $e(k) = tick$. Using $z_e(k)$, $k \in \{0, ..., H\}$, $c(k,j)$ is then given by

$$c(k,j) = \sum_{i=k+1}^{j} z_e(i) \tag{8}$$

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

Then, $z_e(k) = 1$ if and only if

$$\alpha^{\mathsf{T}} w(k-1) = 1 \wedge \beta^{\mathsf{T}} w(k) = 1.$$

Thus, $z_e(k) = 1$ if and only if the following equations hold.

$$z_e(k) \leq \alpha^{\mathsf{T}} w(k-1), \tag{9}$$

$$z_e(k) \leq \beta^{\mathsf{T}} w(k) \tag{10}$$

$$z_e(k) \geq -1 + \alpha^{\mathsf{T}} w(k-1) + \beta^{\mathsf{T}} w(k). \tag{11}$$

4.3 Encoding the ticked LTL_f formula

We introduce $H+1$ binary variables $z_\phi(k) \in \{0,1\}$ for $k \in \{0, 1, ..., H\}$, such that $z_\phi(k) = 1$ if and only if $\pi(k...)$ satisfies ϕ. The encodings for the ticked LTL_f formula ϕ 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 ϕ is encoded as follows:

$$v^{\mathsf{T}} w(k) \geq z_\phi(k), \tag{12}$$

$$v^{\mathsf{T}} w(k) < z_\phi(k) + 1. \tag{13}$$

(negation): Let $\phi = \neg\psi$. Then, the satisfaction of ϕ is encoded as

$$z_\phi(k) = 1 - z_\psi(k). \tag{14}$$

(conjunction): Let $\phi = \bigwedge_{\ell=1}^{L} \psi_\ell$. Then,

$$z_\phi(k) \leq z_{\psi_\ell}(k),\ \forall \ell \in \{1, ..., L\},$$

$$z_\phi(k) \geq 1 - L + \sum_{\ell=1}^{L} z_{\psi_\ell}(k). \tag{15}$$

(disjunction): Let $\phi = \bigvee_{\ell=1}^{L} \psi_\ell$. Then,

$$z_\phi(k) \geq z_{\psi_\ell}(k),\ \forall \ell \in \{1, ..., L\},$$

$$z_\phi(k) \leq \sum_{\ell=1}^{L} z_{\psi_\ell}(k). \tag{16}$$

With a slight abuse of notation, Boolean operators are used for binary variables. For example, when we consider $\phi = \bigwedge_{\ell=1}^{L} \psi_\ell$, we write $z_\phi = \bigwedge_{\ell=1}^{L} z_{\psi_\ell}$ instead of (15). Then, we describe the translation of the temporal operator *until* with this notation.

(until): Let $\phi = \psi_1 U_{[m,n]} \psi_2$. We introduce binary variables $\underline{z}_c(k, j), \overline{z}_c(k, j) \in \{0, 1\}$, for $k, j \in \{0, ...H\}$ with $k \leq j$, such that $\underline{z}_c(k, j) = 1$ (resp. $\overline{z}_c(k, j) = 1$) if and only if $m \leq c(k, j)$ (resp. $c(k, j) \leq n$). That is, $c(k, j)$ is encoded as

$$m - M \leq c(k, j) - M\underline{z}_c(k, j) < m \tag{17}$$

$$n < c(k, j) + M\overline{z}_c(k, j) \leq n + M, \tag{18}$$

where M is a sufficiently large number satisfying $M > n$. Then, the satisfaction of ϕ is encoded as

$$z_\phi(k) = \bigvee_{j=k}^{H} z_\phi(k, j), \tag{19}$$

where

$$z_\phi(k, j) = \overline{z}_c(k, j) \wedge \underline{z}_c(k, j) \wedge z_{\psi_2}(j) \wedge \left(\bigwedge_{\ell=k}^{j-1} z_{\psi_1}(\ell) \right).$$

The encodings for $\Diamond_{[m,n]}$ and $\Box_{[m,n]}$ are similarly done from the relation (6) and are thus omitted.

Example 3. We consider a ticked LTL_f formula $\phi = (ap_1 \wedge ap_2) \vee ap_3 = \phi_1 \vee ap_3$ where $ap_1, ap_2, ap_3 \in AP$. ϕ is translated into the following ILP constraints.

$$\begin{aligned}
&z_\phi(k) \geq z_{\phi_1}(k), z_\phi(k) \geq z_{ap_3}(k),\\
&\quad z_\phi(k) \leq z_{\phi_1}(k) + z_{ap_3}(k),\\
&z_{\phi_1}(k) \leq z_{ap_1}(k), z_{\phi_1}(k) \leq z_{ap_2}(k),\\
&\quad z_{\phi_1}(k) \geq 1 - 2 + z_{ap_1}(k) + z_{ap_2}(k),\\
&(v^1)^T w(k) \geq z_{ap_1}(k), (v^1)^T w(k) < z_{ap_1} + 1,\\
&(v^2)^T w(k) \geq z_{ap_2}(k), (v^2)^T w(k) < z_{ap_2} + 1,\\
&(v^3)^T w(k) \geq z_{ap_3}(k), (v^3)^T w(k) < z_{ap_3} + 1,
\end{aligned}$$

where $v^n \in \{0,1\}^N$ is a binary vector such that $v_i^n = 1$ (the i-th component of v^n is 1) if and only if $ap_n \in L(s_i)$ for $n \in \{1, 2, 3\}$.

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 ϕ.

$$\text{find} : \begin{cases} w(k), z_\phi(k), k \in \{0, ..., H\}, & (20)\\ z_e(k), k \in \{1, ..., H\}, & (21)\\ c(k, j), k, j \in \{0, ..., H\}, k \leq j, & (22) \end{cases}$$

subject to the following constraints:

$$(7) - (8),\ ILP(\phi),\ z_\phi(0) = 1, \tag{23}$$

where $ILP(\phi)$ is the constraints of an ILP problem for ticked LTL_f formula ϕ 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 Bjorner (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 $move_{ij}$ and $reach_{ij}$, $i, j \in \{1, \ldots, 4\}$ represent the events that are

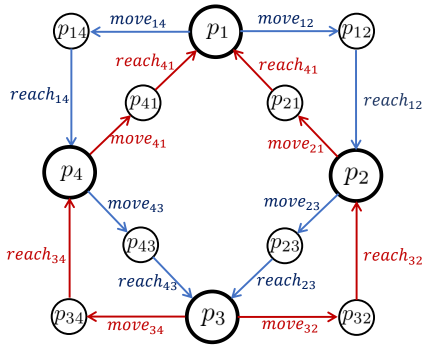


Fig. 2. The untimed transition system G_{act} considered in the simulation example.

associated with the edges. More specifically, the event $move_{ij}$ indicates that the agent decides to move from p_i to p_j, and the event $reach_{ij}$ 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 $L_{act}(p_i) = \{ap_i\}$, $\forall i \in \{1, ...4\}$. The initial state is p_1.

The timer interval T_σ is then defined as follows: if $reach_{i,j}$ for $i, j \in \{1, \ldots, 4\}$ is defined in Fig. 2, $T_{reach_{ij}}$ is given by

$$T_{reach_{12}} = [2, \infty], T_{reach_{14}} = [1, \infty], T_{reach_{21}} = [3, \infty], \\ T_{reach_{23}} = [2, \infty], T_{reach_{32}} = [3, \infty], T_{reach_{34}} = [2, \infty], \\ T_{reach_{41}} = [2, \infty], T_{reach_{43}} = [1, \infty]. \quad (24)$$

For example, $T_{reach_{12}} = [2, \infty]$ 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 $reach_{12}$ can occur at any time after 2 ticks. In other words, the agent takes at least 2 ticks to move from p_i to p_j. On the other hand, if $move_{ij}$ for $i, j \in \{1, \ldots, 4\}$ is defined in Fig. 2, $T_{move_{ij}}$ is then given by

$$T_{move_{ij}} = [0, \infty]. \quad (25)$$

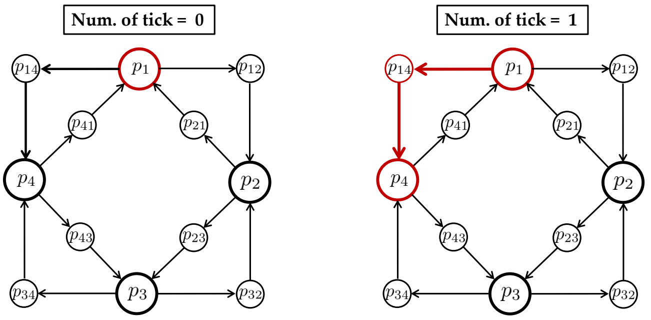
(25) indicates that, if the state of the agent is s_i, the event $move_{ij}$ can occur at any time (i.e., after any number of the event *tick* occurs). Based on the above definitions, the corresponding timed transition system G is constructed according to Definition 2.

5.2 Simulation results

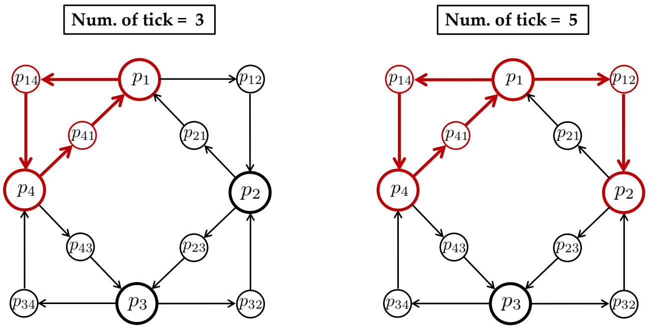
We first consider the following specification: $\phi_1 = \Diamond_{[1,5]} ap_2 \wedge \Diamond_{[1,5]} ap_4$. That is, starting from the initial position (i.e., p_1), the agent must reach p_2 and p_4 while the number of the event *tick* is between 1 and 5. The corresponding ILP problem is solved with different selections of H, in order to find the execution satisfying ϕ_1. Specifically, starting from $H = 5$, we solve the corresponding ILP problem and we increment the horizon until the execution satisfying ϕ_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 *tick* given by 1 (right figure of Fig. 3(a)), and $\Diamond_{[1,5]} ap_2$ is satisfied with the total number of *tick* given by 5 (right figure of Fig. 3(b)). The result of execution is concretely given by

$$\pi_1 = p_1, move_{14}, p_{14}, tick, p_{14}, reach_{14}, p_4, ... \\ move_{41}, p_{41}, tick, p_{41}, tick, p_{41}, reach_{41}, p_1, ... \\ move_{12}, p_{12}, tick, p_{12}, tick, p_{12}, reach_{12}, p_2. \quad (26)$$

Therefore, the result of execution is shown to satisfy ϕ_1. (26) implies that the agent aims to satisfy ap_4 and then



(a) Partial execution of π_1 until the number of the event *tick* is 1.



(b) Partial execution of π_1 until the number of the event *tick* is 5.

Fig. 3. Result of execution π_1 by solving the ILP problem. In the figure, red nodes and edges represent the path that the agent traverses according to π_1.

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 ϕ_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 *tick* being from 3 to 5.

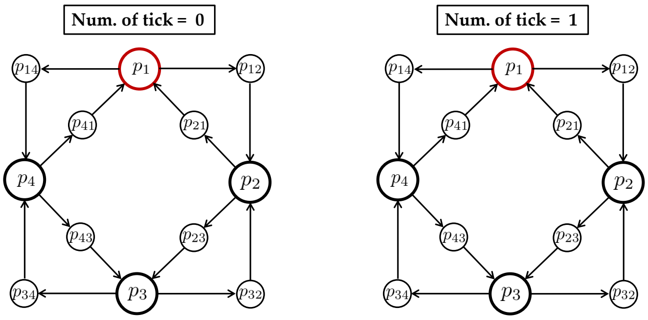
The execution satisfying ϕ_2 is found with $H = 10$ and the result is shown in Fig. 4. The figure shows that the agent reaches p_3 while avoiding p_2 with total number of *tick* given by 3 (Fig.4(b)). The result of execution is concretely given by

$$\pi_2 = p_1, tick, p_1, move_{14}, p_{14}, tick, p_{14}, reach_{14}, p_4, ... \\ move43, p_{43}, tick, p_{43}, reach_{43}, p_3, move_{32}, p_{32}, ... \\ tick, p_{32}. \quad (27)$$

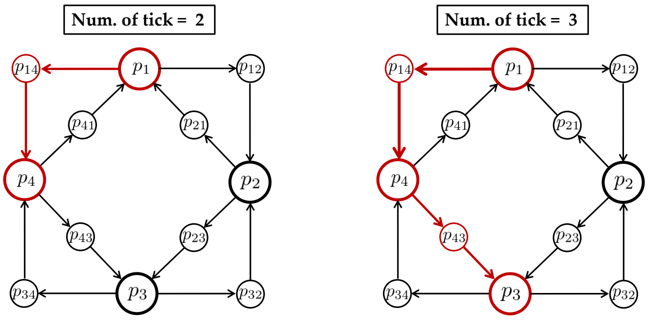
Therefore, it is shown that the agent satisfies the formula ϕ_2.

6. CONCLUSION AND FUTURE WORK

In this paper, we considered a TDES proposed by Brandin and Wonham, where the elapse of time is described by an event *tick*, and proposed ticked LTL$_f$ that describes real-time constraints based on the occurrence of *tick* in the TDES. To find the solution of Problem 1 we provided an approach to encode Problem 1 into ILP. Then, we illus-



(a) Partial execution of π_2 until the number of the event *tick* is 1.



(b) Partial execution of π_2 until the number of the event *tick* is 3.

Fig. 4. Result of execution π_2 by solving the ILP problem. In the figure, red nodes and edges represent the path that the agent traverses according to π_2.

trated the effectiveness of the proposed approach through a numerical example.

Note that this paper deals with the problem of finding a *feasible* execution sequence of TDES, such that the ticked LTL_f is satisfied. Hence, future work involves finding an *optimal* execution sequence, such that a certain cost function is minimized while the ticked LTL_f is satisfied.

ACKNOWLEDGEMENT

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

REFERENCES

C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems, 2nd edition*. Springer, 2008.

J. Campos, C. Seatzu, and X. Xie. *Formal methods in Manufacturing*, CRC Press, 2014.

C. Seatzu, M. Silva, and J. H. van Schuppen(Eds.). *Control of Discrete Event Systems Automata and Petri Net Perspective, Letcure Notes in Control and Information Science*. vol. 433, Springer, 2013.

J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg(Eds.). *Real-Time:Theory and Practice. Lecture Notes in Computer Science*, vol. 600. Springer, 1991

R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, vol. 126, pages 139–152, 1996.

J. F. Raskin and P. Y. Schobbens. Real-time logics: Fictitious clock as an abstraction of dense time. *Lecture Notes in Computer Science*, vol. 1217. pages 165-182, Springer, 1997.

T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *19th International Colloquium on Automata, Languages, and Programming*, pages 545-558, 1992.

J. S. Ostroff and W. M. Wonham. A framework for real-time discrete event control. *IEEE Trans. on Automatic Control*, vol. 35, no. 4, pages 386–397, 1990.

B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. on Automatic Control*, vol. 39, no. 2, pages 329–342, 1994.

C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

E. M. Clarke,Jr., O. Grumberg, D. Kroening, D. Peled, and H. Veith. *Model Checking, end Edition*, MIT Press, 2018.

A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, vol. 2, no. 5, pages 1–64, 2006.

J. Thistle and W. Wonham. Control problems in a temporal logic framework. *Int. J. Control*, vol. 44, no. 4, pages 943-976, 1986.

S. Jiang and R. Kumar. Supervisory control of discrete event systems with CTL* temporal logic specifications. *SIAM J. Control Optim.*, vol. 44, no. 6, pages 2079-2103, 2006.

A. Sakakibara and T. Ushio. Decentralized supervision and coordination of concurrent discrete event systems under LTL constraints, In *Proc. 14th Workshop on Discrete Event Systems*, pages 18–23, 2018.

C. Belta, B. Yordanov, and E. A. Gol. *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic -based reactive mission and motion planning. *IEEE Trans. Robotics*, vol. 25, no. 6, pages 1370–1381, 2009.

T. Wongpiromsam, U. Topcu, R. M. Murray. Receding horizon temporal logic planning. *IEEE Trans. Automat. Contr.*, vol. 57, no. 11, pages 2817–2830, 2012.

S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi. Symbbolic LTL_f synthesis. In *Proc. 26th Int. Joint Conf. Artificial Intelligence*, pages 1362–1369, 2017.

J. Li, K. Y. Rozier, G. Pu, Y. Zhang, and M. Y. Vardi. SAT-based explict LTL_f satisfiability checking. In *Proc. 33rd AAAI Conf. on Artificial Intelligence*, pages 2946–2953, 2019.

R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, vol. 2, pages 255–299, 1990.

O. Maler and D. Nickovi. Monitoring temporal properties of continuous signals. *Lecture Notes in Computer Science*, vol. 3253, pages 152–166, 2004.

J. S. Ostroff. Deciding properties of timed transition models. *IEEE Trans. Parellel and Disributed Systems*, vol. 1, no. 2, pages 170–183, 1990.

M. Barbeau, F. Kabanza, and R. St.-Denis. A method for the synthesis of controllers to handle safety, liveness, and real-time constraints. *IEEE Trans. Automat. Contr.*, vol. 43, no. 11, pages 1543–1559, 1998.

A. Dhananjayan and K. T. Seow. A metric temporal logic specification interface for real-time discrete-event control. *IEEE Trans. Systems, Man & Cybernetics*, vol. 44, no. 9, pages 1204–1215, 2014.

L. Moura and N. Bjørner. Z3: An Efficient SMT Solver. *Lecture Notes in Computer Science*, vol. 4963, pages 337–340, Springer, 2008.

E. M. Wolff, U. Topcu, and R. M. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *IEEE International Conference on Robotics and Automation*, pages 5319–5325, 2014.