Hyper-Labeled Transition System and its Application to Planning under Linear Temporal Logic Constraints

Takuma Kinugawa, Toshimitsu Ushio, *Member, IEEE*

***Abstract*— Recently, formal methods have been paid much attention to in planning. We often leverage a labeled transition system (LTS) with a set of atomic propositions as a model of an environment. However, for example, in a flexible manufacturing system where an assembling machine has several functions that are performed exclusively, we need several states labeled by different sets of atomic propositions to discriminate the functions explicitly. This paper aims to reduce the number of the states. We introduce an extension of the LTS, called a *hyperLTS*, the labeling function of which assigns a set of sets of atomic propositions to each state. Then, we propose linear encodings for the hyperLTS to represent a sequence of pairs of a state and a selected set of atomic propositions. The hyperLTS-based modeling is consequently applied to a planning problem with one hard constraint and several soft constraints, thereby converting it into an integer linear programming problem. The effectiveness of the proposed modeling is illustrated through an example of a path planning problem of a mobile robot in a manufacturing system.**

***Index Terms*— Formal method, optimal control, modeling, linear temporal logic.**

I. Introduction

In formal methods, wherein mathematically rigorous techniques are used for the verification of software and hardware systems so as to make the system reliable and robust, software systems are often modeled by a labeled transition system (LTS) where a labeling function assigns a set of atomic propositions to each state [1]–[3]. Moreover, in model checking, a linear temporal logic (LTL) formula is often used for specifying a desired complex behavior of the system since it is familiar with natural languages and the verification of the software system is done using the LTS and the LTL formula [4].

Recently, formal methods have been applied to control and planning, where the LTS is leveraged as a model of an environment and a specification is described by an LTL formula [5]. There are primarily two approaches to synthesize controllers and planners. One is a game-theoretic approach using automata [6], and the other is an optimization-based approach using linear encodings [7]. The game-theoretic approach is useful for the synthesis of feedback controllers while the optimization-based approach is for that of open-loop controllers and applied to model predictive control. Furthermore, the optimization-based method is often used for planning problems that are converted into integer linear programming (ILP) or mixed-integer linear programming (MILP) problems [8]–[12].

In a planning problem with specified initial and final locations, a desired path is not an infinite sequence of locations but a finite one. To describe a specification for a finite path, LTL over finite traces, named LTL_f, was proposed [13] and synthesis methods based on LTL_f formulas have been studied [14], [15]. Since LTL_f is defined over a finite trace with the same expressiveness of LTL, LTL_f formulas have been recently used as specifications for planning problems [16].

In a planning problem with a specification given by an LTL/LTL_f formula, the satisfaction of the LTL/LTL_f formula by a trace is based on the set of atomic propositions at each state in the trace, that is, each atomic proposition is analyzed to determine whether it belongs to the set. For example, we consider the planning problem of a mobile robot in a manufacturing system where assembling machines have several functions that are performed exclusively, that is, several functions that can not be performed simultaneously. Then, we have to determine not only the movement of the robot to the machine but also the function the robot performs. For this purpose, a set of states in an LTS modeling the system is defined, in which some state in the model is represented by a pair of machine's location and function. Thus, when the number of functions that are performed exclusively increases, the number of states increases because a labeling function is defined to assign a set of atomic propositions to each state uniquely.

In this paper, to reduce the number of states of the LTS, we propose a novel labeling function, called a *hyper-labeling function*, and the encoding for the behaviors of an LTS with its hyper-labeling function, called a *hyper-labeled transition system* or a *hyperLTS* for short. Then, we consider planning problems for hyperLTSs with specifications described by LTL_f formulas and we convert the problems into ILP problems. Finally, we show that the ILP problems can be efficiently solved using the hyperLTSs.

The rest of this paper is organized as follows. In Section II, we review the definition of LTSs and LTL_f. In Section III, we introduce a hyper-labeling function and a hyperLTS. In Section IV, we consider a planning problem with a hyperLTS. Then, we propose novel linear encoding for behaviors of a hyperLTS and convert the problem into ILP problems.

The authors are with the Graduate School of Engineering Science, Osaka University, Osaka, Japan (e-mail: kinugawa@hopf.sys.es.osaka-u.ac.jp, ushio@sys.es.osaka-u.ac.jp).

This work partially supported by JST ERATO Grant Number JPMJER1603 and JST CREST Grant Number JPMJCR2012.

We discuss efficiency of the hyperLTS-based modeling by comparing it with the conventional LTS-based modeling for the number of binary variables and the time for encodings. In Section V, as an example, we consider a path planning problem of a mobile robot in a manufacturing system. In Section VI, we provide a summary of the paper and future research directions.

II. Preliminary

Notation : For integers m and n with $m \leq n$, $[m,\ n]$ denotes a set of integers between m and n, that is, $[m,\ n] = \{m, m+1, \ldots, n\}$. For a set A, denoted by $|A|$ is its cardinality. Let $\mathbb{N}_{\geq 0}$ be the set of non-negative integers.

First, we review an LTS and LTL$_f$.

Definition 1: A labeled transition system (LTS) is defined by a tuple $\mathcal{T} = (S, \delta, s_{init}, AP, L)$ where S is a set of states, $\delta \subseteq S \times S$ is a transition relation, $s_{init} \in S$ is the initial state, AP is a set of atomic propositions, and $L : S \rightarrow 2^{AP}$ is a labeling function. □

A *finite execution* π with the length $\mathsf{L}+1$ of an LTS is a finite sequence of states $\pi = s(0)s(1)\ldots s(\mathsf{L}) \in S^{\mathsf{L}+1}$ where $\mathsf{L} \in \mathbb{N}_{\geq 0}$, $s(k) \in S$ with $s(0) = s_{init}$ for all $k \in [0,\ \mathsf{L}]$, and $(s(k'), s(k'+1)) \in \delta$ for all $k' \in [0,\ \mathsf{L}-1]$. For given π and $k \in [0,\ \mathsf{L}]$, $\pi(k...) = s(k)s(k+1)\ldots s(\mathsf{L})$ denotes the k-th suffix of π. A finite sequence of sets of atomic propositions $\mu = p(0)p(1)\ldots p(\mathsf{L}) \in (2^{AP})^{\mathsf{L}+1}$ is called a *trace*. For μ, $\mu(k\ldots) = p(k)p(k+1)\ldots p(\mathsf{L})$ denotes the k-th suffix of μ. We call a trace $\mu_\pi = L(s(0))L(s(1))\ldots L(s(\mathsf{L}))$ *a trace of an execution* $\pi = s(0)\ldots s(\mathsf{L})$ *of* $\mathcal{T}$.

Next, we review the syntax and the semantics of LTL$_f$.

Definition 2 (Syntax of LTL$_f$): A linear temporal logic over finite traces (LTL$_f$) formula is recursively defined by the following grammar.

$$\varphi := True \mid ap \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2\ ,$$

where φ, φ_1, and φ_2 are LTL$_f$ formulas and ap is an atomic proposition. □

$\mathcal{U}$ is the temporal operator called the *until* operator. Additionally, the *eventually* operator, denoted by $\mathcal{F}$, and the *globally* operator, denoted by $\mathcal{G}$, are given by $\mathcal{F}\varphi = True\mathcal{U}\varphi$ and $\mathcal{G}\varphi = \neg(\mathcal{F}\neg\varphi)$, respectively.

The semantics of LTL$_f$ is defined over a finite trace as follows.

Definition 3 (Semantics of LTL$_f$): Given a finite trace $\mu = p(0)p(1)\ldots p(\mathsf{L})$, the satisfaction of an LTL$_f$ formula φ for the k-th suffix of μ ($k \in [0,\ \mathsf{L}]$), denoted by $\mu(k...) \models \varphi$, is defined recursively as follows.

$$\begin{aligned}
\mu(k...) &\models True,\\
\mu(k...) &\models ap \text{ if and only if } ap \in p(k),\\
\mu(k...) &\models \neg\varphi \text{ if and only if } \mu(k...) \not\models \varphi,\\
\mu(k...) &\models \varphi_1 \wedge \varphi_2 \text{ if and only if } \mu(k...) \models \varphi_1 \wedge \mu(k...) \models \varphi_2,\\
\mu(k...) &\models \bigcirc\varphi \text{ if and only if } k+1 \leq \mathsf{L} \wedge \mu(k+1...) \models \varphi,\\
\mu(k...) &\models \varphi_1\mathcal{U}\varphi_2 \text{ if and only if } \exists k' \in [k,\ \mathsf{L}] \text{ s.t.,}\\
&\qquad \mu(k'...) \models \varphi_2 \wedge (\mu(k''...) \models \varphi_1,\ \forall k'' \in [k,\ k'-1]).
\end{aligned}$$

□

A trace μ satisfies φ, denoted by $\mu \models \varphi$, if and only if $\mu(0...) \models \varphi$.

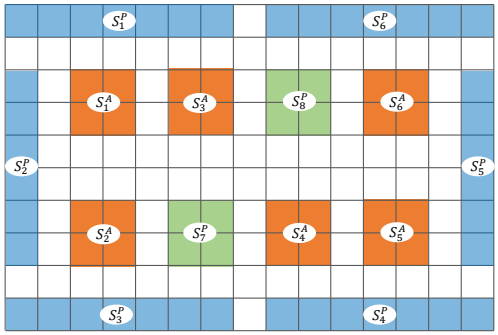

Fig. 1. Grid model of the area where the robot moves. The left-top location is s_0^+, the right-top location is s_{14}^+, the left-bottom location is s_{135}^+, and the right-bottom location is s_{149}^+. There are machines for assembling products at orange states. In conditions favorable to hyperLTSs (*resp*. LTSs), there are parts for products at blue states (*resp*. blue and green states).

III. Hyper-labeled transition system

The labeling function L assigns uniquely a set of atomic propositions to each state in the LTS $\mathcal{T} = (S, \delta, s_{init}, AP, L)$. However, it is practically preferred to be able to select the assignment of atomic propositions to each state at every time. For example, we consider a mobile robot that can move to the place of the machine that can assemble two kinds of products exclusively. We model the working space by a grid and assign the cell with the machine to two situations: "the robot stays and can assemble Product1" and "the robot stays and can assemble Product2". Then, we select one of the assignments exclusively depending on the mission of the robot. See Example 1 below for its detail. We call such an extension of assignments *hyper-labeling for assignments of atomic propositions* or simply *hyper-labeling*. To represent hyper-labeling, we introduce a *hyper-labeling function* $L^+ : S^+ \rightarrow 2^{2^{AP}}$ where S^+ is a set of states and a transition system with hyper-labeling functions, called a *hyper-labeled transition system* (hyperLTS), which is a generalization of the LTS. A set of atomic propositions that are true at each state is not determined uniquely but selected from the assignments given by the hyper-labeling function. We give the formal definition of a hyperLTS, denoted by $\mathcal{T}^+$, as follows.

Definition 4: A hyper-labeled transition system (hyperLTS) is defined by a tuple $\mathcal{T}^+ = (S^+, \delta^+, s_{init}^+, AP, L^+)$ where S^+ is a set of states, $\delta^+ \subseteq S^+ \times S^+$ is a transition relation, $s_{init}^+ \in S^+$ is the initial state, AP is a set of atomic propositions, and $L^+ : S^+ \rightarrow 2^{2^{AP}}$ is a hyper-labeling function. □

We consider an example with a hyperLTS.

Example 1: We consider a mobile robot in a manufacturing system with assembling machines. The area where the robot moves is represented by a 10×15 grid, as shown in Fig. 1. The hyperLTS is given by $\mathcal{T}^+ = (S^+, \delta^+, s_{init}^+, AP, L^+)$ where $S^+ = \{s_0^+, s_1^+, \ldots, s_{149}^+\}$ and $(s_i^+, s_j^+) \in \delta^+$ if and only if s_i^+ and s_j^+ are adjacent states horizontally or vertically. At each orange state, there is a machine which can assemble products. Denoted by S_1^A is a set of four states in Fig. 1. At each state in S_1^A, there is the machine which can assemble two products: Product1 and Product2. Let $Assemble_1$ and $Assemble_2$ be

atomic propositions which indicate "assembling Product1" and "assembling Product2", respectively. It is considered that the machine assembles them exclusively. Thus, both $Assemble_1$ and $Assemble_2$ are not true at the same time. To represent this assignment for $s \in S_1^A$, $L^+(s)$ is given by

$$L^+(s) \quad = \quad \{\emptyset,\ \{Assemble_1\},\ \{Assemble_2\}\}.$$

Whenever the robot stays at $s \in S_1^A$, we select one set from $L^+(s)$. □

A finite execution π and trace μ of a hyperLTS $\mathcal{T}^+$ are defined as the same way as those of an LTS. We call a trace $\mu_\pi^+ = p_\pi(0)p_\pi(1)\dots p_\pi(\mathsf{L})$ *a trace of an execution* $\pi = s(0)\dots s(\mathsf{L})$ *of* $\mathcal{T}^+$ if and only if $p_\pi(k) \in L^+(s(k))$ for each $k \in [0,\ \mathsf{L}]$. Note that, even if $s(j) = s(k)$ in an execution π of a hyperLTS for $j, k \in [0,\ \mathsf{L}]$, $j \neq k$, the atomic propositions that hold at $s(j)$ and $s(k)$ may be different, i.e., $p_\pi(j) \neq p_\pi(k)$ in a trace of π. Then, we say that π has a trace satisfying an LTL_f formula φ if and only if there exists a trace μ_π^+ of π such that $\mu_\pi^+ \models \varphi$.

IV. Planning using a hyperLTS

In this section, we formulate a planning problem where an environment is modeled by a hyperLTS and a specification is described by LTL_f formulas. Then, we solve the problem by converting it into ILP problems.

A. Problem formulation

We consider a planning problem for a hyperLTS with a specification consisting of one hard constraint and several soft constraints, where the hard constraint is a mandatory requirement and each soft constraint is an optional requirement preferable to be satisfied. The hard constraint is denoted by ϕ and the soft constraints are denoted by $\psi_1, ..., \psi_{N_S}$ where N_S denotes the number of soft constraints. We assume that there is a preference for each soft constraint, that is, if there is no trace that satisfies both ψ_i and ψ_j but there are traces that satisfy one of them, then, we select a trace that satisfies the more preferable constraint. To represent the preference, we introduce a positive integer W_n, called a *weight*, for each soft constraint $\psi_n (n \in [1,\ N_S])$ and interpret that ψ_n is more preferable than ψ_m if $W_n > W_m$. Thus, a trace for which the sum of the weights of the satisfied soft constraints is larger is a more preferable one. We denote a set of pairs of a soft constraint and its weight by $\Psi = \{(\psi_n, W_n) |\ n \in [1,\ N_S]\}$. Then, the problem considered in this paper is formulated as follows.

Problem 1: Given a hyperLTS $\mathcal{T}^+$, a hard constraint ϕ, a set of pairs of a soft constraint and its weight Ψ, and a positive integer L, find a trace μ_π^+ of an execution π with the length $\mathsf{L}+1$ of $\mathcal{T}^+$ such that

maximize:	the sum of the weights of soft constraints that μ_π^+ satisfies.
subject to:	$\mu_\pi^+ \models \phi$,
	μ_π is a trace of an execution π of $\mathcal{T}^+$.

□

There are two approaches to solve Problem 1. One is that the hyperLTS is used directly. The other is that the LTS induced by the given hyperLTS is used instead of the hyperLTS. When the hyperLTS $\mathcal{T}^+$ is given by $\mathcal{T}^+ = (S^+, \delta^+, s_{init}^+, AP, L^+)$ with $|S^+| \geq 1$, its induced LTS is given by $\mathcal{T} = (S, \delta, s_{init}, AP, L)$, where $S = \bigcup_{s^+ \in S^+} \{s^+\} \times L^+(s^+)$, $(s_1, s_2) \in \delta$ for $s_1 = (s_1^+, P_1)$ and $s_2 = (s_2^+, P_2)$ with $(s_1^+, s_2^+) \in \delta^+$, $P_1 \in L^+(s_1^+)$, and $P_2 \in L^+(s_2^+)$, $s_{init} = (s_{init}^+, P)$ with $P \in L^+(s_{init}^+)$, $L(s) = P$ for $s = (s^+, P) \in S$ and $P \in L^+(s^+)$.

In each approach, we solve Problem 1 by converting it into an ILP problem. For this purpose, we introduce a method to encode a finite execution and its trace, and the satisfaction of an LTL_f formula into sets of equations that are constraints in the ILP problem.

B. Encoding an execution and its trace

First, we consider the encoding of a finite execution of a hyperLTS $\mathcal{T}^+$, where $s_{init}^+ \in S^+ = \left\{s_1^+, s_2^+ \dots, s_{|S|}^+\right\}$. Let $\pi = s(0)s(1)\dots s(\mathsf{L})$ be a finite execution with the length $\mathsf{L}+1$ of $\mathcal{T}^+$ and $s(0) = s_{init}^+$. We introduce $\mathsf{L}+1$ binary vectors $\boldsymbol{w}(k) = [w_1(k), w_2(k), \dots, w_{|S^+|}(k)]^\mathsf{T} \in \{0,1\}^{|S^+|}$ for $k \in [0,\ \mathsf{L}]$ to represent the k-th state $s(k)$ of π as follows.

$$w_i(k) = \begin{cases} 1 & \text{if } s_i^+ = s(k), \\ 0 & \text{otherwise.} \end{cases}$$

Since $s(0) = s_{init}^+$, we have $w_i(0) = 1$ if and only if $s_i^+ = s_{init}^+$. Denoted by $\boldsymbol{A} \in \{0,1\}^{|S^+| \times |S^+|}$ is the *transition matrix* of $\mathcal{T}^+$, where the (i, j)-th element $A_{i,j}$ of $\boldsymbol{A}$ is defined by

$$A_{i,j} = \begin{cases} 1 & \text{if } (s_i^+, s_j^+) \in \delta^+, \\ 0 & \text{otherwise.} \end{cases}$$

This provides the following encoding.

$$\boldsymbol{w}(k+1) \leq \boldsymbol{A}^\mathsf{T}\boldsymbol{w}(k),\ \mathbf{1}_{|S^+|}^\mathsf{T}\boldsymbol{w}(k) = 1, \tag{1}$$

where $\mathbf{1}_M$ is the M-dimensional vector, all the elements of which are 1. Similarly, a finite execution of an LTS $\mathcal{T}$ is encoded by (1) with $\boldsymbol{w'} \in \{0,1\}^{|S|}$ and $\boldsymbol{A'} \in \{0,1\}^{|S| \times |S|}$.

Next, we consider the encoding of a finite trace of the execution π. Since the hyper-labeling function assigns a set of sets of atomic propositions to each state, we introduce $\mathcal{P} \subseteq 2^{AP}$ as follows.

$$\mathcal{P} = \{P_1, P_2, \dots, P_{|\mathcal{P}|}\} = \bigcup_{s^+ \in S^+} L^+(s^+).$$

Let $\mu_\pi^+ = p_\pi(0)p_\pi(1)\dots p_\pi(\mathsf{L})$ be a trace of the execution π of $\mathcal{T}^+$. We introduce $\mathsf{L}+1$ binary vectors $\boldsymbol{t}(k) = [t_1(k), t_2(k), \dots, t_{|\mathcal{P}|}(k)]^\mathsf{T} \in \{0,\ 1\}^{|\mathcal{P}|}$ for $k \in [0,\ \mathsf{L}]$ to represent the k-th set of atomic propositions $p_\pi(k)$ of μ_π^+ as follows.

$$t_i(k) = \begin{cases} 1 & \text{if } P_i = p_\pi(k), \\ 0 & \text{otherwise.} \end{cases}$$

We introduce the *labeling matrix* of $\mathcal{T}^+$, denoted by $\boldsymbol{V} \in \{0,\ 1\}^{|\mathcal{P}| \times |S^+|}$, where the $(i,\ j)$-th element $V_{i,j}$ of $\boldsymbol{V}$ is defined by

$$V_{i,j} = \begin{cases} 1 & \text{if } P_i \in L^+(s_j^+), \\ 0 & \text{otherwise.} \end{cases}$$

This provides the following encoding.

$$\boldsymbol{t}(k) \leq \boldsymbol{V}\boldsymbol{w}(k),\ \mathbf{1}_{|\mathcal{P}|}^{\mathsf{T}}\boldsymbol{t}(k) = 1. \tag{2}$$

Note that, in LTSs, we do not need the encoding of a finite trace of the execution since the assignment of a subset of atomic propositions for each state is unique.

C. Encoding the satisfaction of an LTL$_f$ formula

For an LTL$_f$ formula φ, we introduce the following binary variables $z_\varphi(k)$ and $y_\varphi(k)$ ($k \in [0,\ \mathsf{L}]$) to encode the satisfaction of φ into a set of equations for μ_π^+ and μ_π, where μ_π is a trace of an execution of $\mathcal{T}$, respectively.

$$z_\varphi(k) = \begin{cases} 1 & \text{if } \mu_\pi^+(k...) \models \varphi, \\ 0 & \text{otherwise,} \end{cases} \quad y_\varphi(k) = \begin{cases} 1 & \text{if } \mu_\pi(k...) \models \varphi, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we encode atomic propositions for these traces.

Atomic proposition for hyperLTS: Let $\varphi = ap \in AP$ and $v^{ap} \in \{0,1\}^{|\mathcal{P}|}$ be a binary vector such that $v_i^{ap} = 1$ (the i-th element of v^{ap} is 1) if and only if $ap \in P_i$ where $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$. Then, the satisfaction of φ is encoded as follows.

$$(v^{ap})^{\mathsf{T}}\boldsymbol{t}(k) = z_\varphi(k). \tag{3}$$

Atomic proposition for LTS: Let $\varphi = ap \in AP$ and $u^{ap} \in \{0,1\}^{|S|}$ be a binary vector such that $u^{ap} = 1$ (the i-th element of u^{ap} is 1) if and only if $ap \in L(s_i)$ where $S = \{s_1, s_2, \ldots, s_{|S|}\}$. Then, the satisfaction of φ is encoded as follows.

$$(u^{ap})^{\mathsf{T}}\boldsymbol{w}'(k) = y_\varphi(k). \tag{4}$$

References [7] and [11] provide details for the encodings of Boolean operators and temporal operators. Denoted by $ILP_1(\varphi)$ and $ILP_2(\varphi)$ are the sets of constraints obtained by encoding φ with (3) and (4), respectively.

D. ILP problems

Based on the encodings for Problem 1, we convert this problem into the following ILP problem **ILP1** (*resp*. **ILP2**) for the approach using the hyperLTS (*resp*. the approach using the LTS).

ILP1 :

$$\begin{aligned} \underset{\boldsymbol{w}(0),\ldots,\boldsymbol{w}(\mathsf{L}),\ \boldsymbol{t}(0),\ldots,\boldsymbol{t}(\mathsf{L})}{\text{maximize}} \quad & : \sum_{n=1}^{N_S} W_n \cdot z_{\psi_n}(0) \\ \text{subject to} \quad & : (1),\ (2),\ ILP_1(\phi), \\ & \quad ILP_1(\psi_1), \ldots, ILP_1(\psi_{N_S}),\ \text{and} \\ & \quad z_\phi(0) = 1. \end{aligned}$$

ILP2 :

$$\begin{aligned} \underset{\boldsymbol{w}'(0),\ldots,\boldsymbol{w}'(\mathsf{L})}{\text{maximize}} \quad & : \sum_{n=1}^{N_S} W_n \cdot y_{\psi_n}(0) \\ \text{subject to} \quad & : (1),\ ILP_2(\phi), \\ & \quad ILP_2(\psi_1), \ldots, ILP_2(\psi_{N_S}),\ \text{and} \\ & \quad y_\phi(0) = 1. \end{aligned}$$

Note that both $\boldsymbol{w}(i)$, $\boldsymbol{t}(i)$, and $\boldsymbol{w}'(i)$ are decision variables of the aforementioned problems.

E. Comparison of two approaches

We consider the effectiveness of the approaches using the hyperLTS and the LTS for the number of binary variables and the computation time for encodings.

First, we compare the number of binary variables for both approaches. For $\mathcal{T}^+$, the number of variables needed to encode an execution and a trace of the execution is $|S^+| \cdot (\mathsf{L}+1)$ and $|\mathcal{P}| \cdot (\mathsf{L}+1)$, respectively. The sum of the number of variables needed to encode an execution and its trace for the hyperLTS is as follows.

$$|S^+| \cdot (\mathsf{L}+1) + |\mathcal{P}| \cdot (\mathsf{L}+1). \tag{5}$$

For $\mathcal{T}$, the number of states is $\sum_{s^+ \in S^+} |L^+(s^+)|$, and the number of variables needed to encode an execution with the length $\mathsf{L}+1$ is as follows.

$$\left(\sum_{s^+ \in S^+} |L^+(s^+)| \right) \cdot (\mathsf{L}+1). \tag{6}$$

Then, we have the following remark from (5) and (6).

Remark 1: If we have

$$|S^+| + |\mathcal{P}| < \sum_{s^+ \in S^+} |L^+(s^+)|, \tag{7}$$

the number of variables for encoding of the approach using the hyperLTS is smaller than that using the LTS. □

Second, we consider the computation time for encodings. From (1) and (2), the order of the computation time to encode an execution and its trace of $\mathcal{T}^+$ is as follows.

$$\begin{aligned} &O\left(|S^+| \cdot |S^+| + |S^+| + |\mathcal{P}| \cdot |S^+| + |\mathcal{P}|\right) \\ &= O\left(|S^+| \cdot (|S^+| + |\mathcal{P}|)\right). \end{aligned} \tag{8}$$

On the other hand, the order of the computation time to encode an execution of $\mathcal{T}$ is as follows.

$$O\left(|S| \cdot |S| + |S|\right) = O\left(|S| \cdot |S|\right). \tag{9}$$

Since $|S^+| \leq |S|$, the computation time to encode an execution and its trace by the approach using the hyperLTS is faster than that using the LTS when (7) holds. Moreover, we discuss the computation time to encode a specification. To encode the atomic propositions, we use the binary variables $\boldsymbol{t}$ and $\boldsymbol{w}$ for the approaches using the hyperLTS and the LTS, respectively. Therefore, the orders of the computation time to encode (3) and (4) are $O(|\mathcal{P}|)$ and $O(|S|)$, respectively. Note that we have

$$|\mathcal{P}| = \left| \bigcup_{s^+ \in S^+} L^+(s^+) \right| \leq \sum_{s^+ \in S^+} |L^+(s^+)| = |S|. \tag{10}$$

Thus, the computation time for the encoding of (3) is faster than that of (4). Since the encoding of all LTL$_f$ formulas except atomic propositions are the same for both approaches, the computation time to encode the hard and the soft constraints using the hyperLTS is also faster than that using the LTS.

V. Illustrative example

In this section, we consider a path planning problem of a mobile robot with a finite horizon such that we assemble twelve products, denoted by Product$i$ for $i \in [1,\ 12]$, while the robot collects parts to be needed for the assembling of these products. Shown in Fig. 1 is a grid model of the area that is modeled by the hyperLTS $\mathcal{T}^+ = (S^+, \delta^+, s^+_{init}, AP, L^+)$ with $s^+_{init} = s^+_0$. We consider two cases where the first case is given such that (7) holds while the second case does not satisfy (7). In the first case, we assume that parts used for Product$(2i-1)$ and Product$(2i)$ are at the states in S^P_i for $i \in [1,\ 6]$ where

$$
\begin{aligned}
&S^P_1 = \{s^+_0, s^+_1, \ldots, s^+_6\}, && S^P_2 = \{s^+_{30}, s^+_{45}, \ldots, s^+_{105}\},\\
&S^P_3 = \{s^+_{135}, s^+_{136}, \ldots, s^+_{141}\}, && S^P_4 = \{s^+_{143}, s^+_{144}, \ldots, s^+_{149}\},\\
&S^P_5 = \{s^+_{44}, s^+_{59}, \ldots, s^+_{104}, s^+_{119}\}, && S^P_6 = \{s^+_8, s^+_9, \ldots, s^+_{14}\}.
\end{aligned}
$$

The robot collects them and assembles the products by using the machines. The machines at the states in S^A_i can assemble Product$(2i-1)$ and Product$(2i)$ where

$$
\begin{aligned}
&S^A_1 = \{s^+_{32}, s^+_{33}, s^+_{47}, s^+_{48}\}, && S^A_2 = \{s^+_{92}, s^+_{93}, s^+_{107}, s^+_{108}\},\\
&S^A_3 = \{s^+_{35}, s^+_{36}, s^+_{50}, s^+_{51}\}, && S^A_4 = \{s^+_{98}, s^+_{99}, s^+_{113}, s^+_{114}\},\\
&S^A_5 = \{s^+_{101}, s^+_{102}, s^+_{116}, s^+_{117}\}, && S^A_6 = \{s^+_{41}, s^+_{42}, s^+_{56}, s^+_{57}\}.
\end{aligned}
$$

For each $i \in [1,\ 12]$, let $Parts_i$ and $Assemble_i$ be atomic propositions indicating that the robot collects parts for Product$i$ and that it assembles Product$i$, respectively. Then, AP is given by

$$
AP = \{Parts_i,\ Assemble_i |\ i \in [1, 12]\}.
$$

We consider the case where the machine assembles them exclusively. Then, every pair of $Assemble_i$ and $Assemble_j$ for $i, j \in [1,\ 12]$ with $i \neq j$ is not true at the same time. Then, for $s^+ \in S^+$ and $i \in [1,\ 6]$, the hyper-labeling function $L^+(s^+)$ is given by

$$
L^+(s^+) = \begin{cases} \{\{Parts_{2i-1}\}, \{Parts_{2i}\}, \{Parts_{2i-1}, Parts_{2i}\}\} & \text{if } s^+ \in S^P_i,\\ \{\emptyset, \{Assemble_{2i-1}\}, \{Assemble_{2i}\}\} & \text{if } s^+ \in S^A_i,\\ \{\emptyset\} & \text{otherwise.} \end{cases}
$$

Based on the above setting, we have

$$
|S^+| + |\mathcal{P}| = 181, \quad \sum_{s^+ \in S^+} |L^+(s^+)| = 276.
$$

In the second case, we have

$$
|S^+| + |\mathcal{P}| = 208, \quad \sum_{s^+ \in S^+} |L^+(s^+)| = 172.
$$

Due to space limitations, we omit detailed assignment of hyper-labeling function.

In the following, we describe the hard constraint and the soft constraints. The hard constraint is given by

$$
\phi = \bigwedge_{i \in [1,\ 12]} \phi^i_a \wedge \bigwedge_{i \in [1,\ 12]} \phi^i_p, \tag{11}
$$

where $\phi^i_a = \mathcal{F}(Assemble_i)$, $\phi^i_p = (\neg Assemble_i)\mathcal{U}(Parts_i)$. The first term of (11) describes the robot has to assemble all products. For each $i \in [1,\ 12]$, ϕ^i_p describes that the robot can assemble Product$i$ after collecting its parts. The set of pairs of a soft constraint and its weight is given by

$$
\Psi = \left\{(\psi^1_{10}, 3),\ (\psi^1_{11}, 2),\ (\psi^1_{12}, 1),\ (\psi^7_1, 1),\ (\psi^7_2, 2),\ (\psi^7_3, 3)\right\},
$$

where, for $(i, j) \in \{(1, 10), (1, 11), (1, 12), (7,\ 1), (7,\ 2), (7,\ 3)\}$,

$$
\psi^i_j \quad = \quad (\neg Assemble_j)\mathcal{U}(Assemble_i).
$$

ψ^i_j describes that assembling Product$i$ is more important than assembling Product$j$. Additionally, from the weights of these soft constraints, it is most important to assemble Product1 (*resp.* Product7) before assembling Product10 (*resp.* Product3). For each approach, we formulate the path planing problems with L=30, 35, 40, 50, and 60 into Problem 1 and convert them into **ILP1** and **ILP2**.

The simulation was run by a machine with AMD Ryzen9 5950X and 128GB memory, and the solver Gurobi [17] was used to find optimal solutions of these ILP problems.

For the approaches using the hyperLTS (Hyper) and that using the LTS (LTS), the number of binary variables (# of variables), the time to encode an execution and its trace, or an execution (Encoding time (behavior)), the time to encode the hard constraint and the soft constraints (Encoding time (specification)), the time to solve the ILP problems (Solving time), and the sum of weights of satisfied soft constraints (Sum of weights) are shown in Table I and II. Note that Table I and II are for the first and second case, respectively. For L=30, 35 and 40 (*resp.* L = 30) in the first case (*resp.* second case), both approaches conclude that there is no feasible solution. For L = 50 in the first case, the approach using the hyperLTS obtains the optimal solution while that using the LTS can not after one hour computation. Shown in Fig. 2 is the optimal execution π for L = 50 in the first case. From π, we have the following trace.

$$
\begin{aligned}
t_\pi(0) &= \{Parts_1, Parts_2\}, & t_\pi(2) &= \{Parts_3, Parts_4\},\\
t_\pi(4) &= \{Assemble_1\}, & t_\pi(8) &= \{Assemble_4\},\\
t_\pi(14) &= \{Parts_5, Parts_6\}, & t_\pi(15) &= \{Parts_6\},\\
t_\pi(17) &= \{Parts_7, Parts_8\}, & t_\pi(19) &= \{Assemble_7\},\\
t_\pi(20) &= \{Assemble_8\}, & t_\pi(26) &= \{Parts_9, Parts_{10}\},\\
t_\pi(28) &= \{Assemble_9\}, & t_\pi(29) &= \{Assemble_{10}\},\\
t_\pi(35) &= \{Parts_{11}, Parts_{12}\}, & t_\pi(37) &= \{Assemble_{12}\},\\
t_\pi(38) &= \{Assemble_{11}\}, & t_\pi(43) &= \{Assemble_5\},\\
t_\pi(44) &= \{Assemble_6\}, & t_\pi(46) &= \{Assemble_2\},\\
t_\pi(50) &= \{Assemble_3\},
\end{aligned}
$$

and $\emptyset$ are assigned to the other states in π. The trace of π satisfies the hard constraint ϕ and the soft constraints other than ψ^7_1. To assemble all products with L = 50, the robot assembles Product1 before it assembles Product7. Thus, ψ^7_1 are not satisfied. To satisfy ψ^7_3, the robot selects not to assemble Product3 on the first stay at the states in S^A_2. In the first case (*resp.* second case), the time to encode an execution and its trace with the approach using the hyperLTS is shorter (*resp.* longer) than that with the approach using the LTS for each L. Additionally, for both cases and all L, the time to encode hard

TABLE I

COMPARISON OF THE APPROACHES USING THE HYPERLTS AND AN LTS IN THE FIRST CASE.

L	# of variables		Encoding time (behavior)[s]		Encoding time (specification)[s]		Solving time[s]		Sum of weights	
	Hyper	LTS	Hyper	LTS	Hyper	LTS	Hyper	LTS	Hyper	LTS
30	10230	13175	199.34	546.34	18.50	104.91	0.24	34.42	–	–
35	11880	15300	217.25	601.48	19.83	115.69	35.55	75.05	–	–
40	13530	17425	252.04	687.58	22.83	131.39	2532.30	3042.81	–	–
50	16830	21675	310.70	868.46	28.14	165.47	570.14	–	11	11
60	20130	25925	376.56	1030.90	33.77	195.10	51.79	371.33	12	12

TABLE II

COMPARISON OF THE APPROACHES USING THE HYPERLTS AND AN LTS IN THE SECOND CASE.

L	# of variables		Encoding time (behavior)[s]		Encoding time (specification)[s]		Solving time[s]		Sum of weights	
	Hyper	LTS	Hyper	LTS	Hyper	LTS	Hyper	LTS	Hyper	LTS
30	11067	9951	240.15	211.47	29.42	67.76	1.50	27.77	–	–
35	12852	11556	259.84	239.72	31.15	76.04	66.65	93.25	6	6
40	14637	13161	296.66	276.80	35.38	87.43	10.02	14.30	12	12
50	18207	16371	360.22	347.56	43.16	108.60	24.80	6.09	12	12
60	21777	19581	434.70	406.28	52.18	127.52	16.70	7.45	12	12

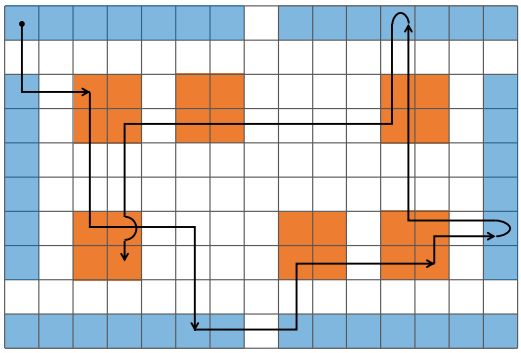

Fig. 2. Result of execution in Section V. The initial state is the left-top state.

and soft constraints with the approach using the hyperLTS is shorter than that for the approach using the LTS. Moreover, in the first case, the solving time of the approach using the hyperLTS is faster than that using the LTS. However, in the second case, the approach using the LTS is more efficient than that using the hyperLTS as L is larger. This result is due to the number of the variables used in the encoding. Thus, (7) is a guideline to leverage the approach using the hyperLTS.

VI. CONCLUSION

We introduced a hyper-labeling function that assigns a set of sets of atomic propositions to each state and defined a novel LTS with the hyper-labeling function, which is called a hyperLTS. We propose linear encodings for both an execution and its trace of a hyperLTS. Then, we formulated a planning problem where an environment is modeled by a hyperLTS and a specification is described by LTL_f formulas. We convert it into ILP problems. As an example, we considered a planning problem of a mobile robot in a manufacturing system where assembling machines have several functions that are performed exclusively.

It is future work to apply the proposed approaches to a hierarchical controller synthesis problem and apply hyperLTSs to a game-theoretic approach.

REFERENCES

[1] A. Arnold, *Finite Transition Systems: Semantics of Communicating Systems*, Prentice Hall, 1996.

[2] M. Müller-Olm, D. Schmidt, and B. Steffen, "Model-checking," In *International Static Analysis Symposium*, pages 330–354, Springer, 1999.

[3] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti, "A state/event-based model-checking approach for the analysis of abstract system properties," *Science of Computer Programming*, vol. 76, no. 2, pages 119–135, 2011.

[4] C. Baier and J.-P. Katoen, *Principles of Model Checking*, MIT press, 2008.

[5] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*, MIT press, 2008.

[6] G. De Giacomo and M. Y. Vardi, "Automata-theoretic approach to planning for temporally extended goals," in *Proceedings of European Conference on Planning*, pages 226–238, Springer, 1999.

[7] 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.

[8] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pages 1372–1395, 2011.

[9] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5319–5325, 2014.

[10] L. Yang and N. Ozay, "Fault-tolerant output-feedback path planning with temporal logic constraints," in *Proceedings of 2018 IEEE Conference on Decision and Control (CDC)*, pages 4032–4039, 2018.

[11] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1189–1206, 2019.

[12] T. Kinugawa, K. Hashimoto, and T. Ushio, "Control of timed discrete event systems with ticked linear temporal logic constraints," *IFAC-PapersOnLine*, vol. 53, no. 2, pages 2143–2148, 2020.

[13] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," In *Proceedings of the 2013 International Joint Conferences on Artificial Intelligence*, pages 854–860, 2017.

[14] S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi, "Symbolic LTL_f synthesis," In *Proceedings of the 2017 International Joint Conferences on Artificial Intelligence*, pages 1362–1369, 2017.

[15] J. Li, K. Y. Rozier, G. Pu, Y. Zhang, and M. Y. Vardi. "Sat-based explicit LTL_f satisfiability checking," In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pages 2946–2953, 2019.

[16] A. Camacho and S. A. McIlraith, "Strong fully observable non-deterministic planning with LTL and LTL_f goals," In *Proceedings of the 2019 International Joint Conferences on Artificial Intelligence*, pages 5523–5531, 2019.

[17] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," Available online: `https://www.gurobi.com`, 2021.