<table>
<thead>
<tr>
<th>Title</th>
<th>Hyper-Labeled Transition System and Its Application to Planning Under Linear Temporal Logic Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Kinugawa, Takuma; Ushio, Toshimitsu</td>
</tr>
<tr>
<td>Citation</td>
<td>IEEE Control Systems Letters. 2022, 6, p. 2437-2442</td>
</tr>
<tr>
<td>Version Type</td>
<td>AM</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://hdl.handle.net/11094/87645">https://hdl.handle.net/11094/87645</a></td>
</tr>
<tr>
<td>rights</td>
<td>© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

*Osaka University Knowledge Archive : OUKA*

https://ir.library.osaka-u.ac.jp/

Osaka University
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.
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| its cardinality. Let \( \mathbb{N}_{\geq 0} \) be the set of non-negative integers.

First, we review an LTS and \( \text{LTL}_f \).

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

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

Next, we review the syntax and the semantics of \( \text{LTL}_f \).

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

\[ \varphi ::= \text{True} \mid ap \mid \neg \varphi_1 \mid \varphi_1 \land \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \cup \varphi_2 , \]

where \( \varphi, \varphi_1, \) and \( \varphi_2 \) are \( \text{LTL}_f \) formulas and \( ap \) is an atomic proposition.

\( \bigcup \) 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 \( \mathcal{F} \varphi = \text{True} \cup \varphi \) and \( \mathcal{G} \varphi = \neg (\mathcal{F} \neg \varphi) \), respectively.

The semantics of \( \text{LTL}_f \) is defined over a finite trace as follows.

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

\[ \begin{align*}
\mu(k) & \models \text{True}, \\
\mu(k) & \models ap \text{ if and only if } ap \in p(k), \\
\mu(k) & \models \neg \varphi \text{ if and only if } \mu(k) \not\models \varphi, \\
\mu(k) & \models \varphi_1 \land \varphi_2 \text{ if and only if } \mu(k) \not\models \varphi_1 \lor \mu(k) \not\models \varphi_2, \\
\mu(k) & \models \bigcirc \varphi \text{ if and only if } \mu(k+1) \models \varphi, \\
\mu(k) & \models \varphi_1 \cup \varphi_2 \text{ if and only if } \exists \mathcal{E} \varphi' \in [k, L] \text{ s.t., } \\
& \quad \mu(k') \models \varphi_2 \land (\mu(k') \models \varphi_1, \forall k' \in [k, K - 1]). \\
\end{align*} \]

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

III. HYPER-LABELED TRANSITION SYSTEM

The labeling function \( L \) assigns uniquely a set of atomic propositions to each state in the LTS \( \mathcal{T} = (S, \delta, s_{\text{init}}, AP, L) \). 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 \( L^+ : S^+ \to 2^{2^{AP}} \) 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 \( \mathcal{T}^+ = (S^+, \delta^+, s_{\text{init}}^+, AP, L^+) \) where \( S^+ \) is a set of states, \( \delta^+ \subseteq S^+ \times S^+ \) is a transition relation, \( s_{\text{init}}^+ \in S^+ \) is the initial state, \( AP \) is a set of atomic propositions, and \( L^+ : S^+ \to 2^{2^{AP}} \) 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 \( 10 \times 15 \) grid, as shown in Fig. 1. The hyperLTS is given by \( \mathcal{T}^+ = (S^+, \delta^+, s_{\text{init}}^+, AP, L^+) \) 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 \( S_{\text{A}}^+ \) is a set of four states in Fig. 1. At each state in \( S_{\text{A}}^+ \), there is the machine which can assemble two products: Product1 and Product2. Let \( \text{Assemble}_1 \) and \( \text{Assemble}_2 \) be
atomic propositions which indicate “assembling Product1” and “assembling Product2”, respectively. It is considered that the machine assembles them exclusively. Thus, both Assemble1 and Assemble2 are not true at the same time. To represent this assignment for $s \in S^+_1$, $L^+(s)$ is given by

$$L^+(s) = \{\emptyset, \{\text{Assemble1}\}, \{\text{Assemble2}\}\}.$$ Whenever the robot stays at $s \in S^+_1$, we select one set from $L^+(s)$.

A finite execution $\pi$ and trace $\mu$ of a hyperLTS $T^+$ are defined as the same way as those of an LTS. We call a trace $\mu^*_s = p_s(0)p_s(1)\ldots p_s(L)$ a trace of an execution $\pi = s(0)\ldots s(L)$ of $T^+$ if and only if $p_s(k) \in L^+(s(k))$ for each $k \in [0, L]$. Note that, even if $s(j) = s(k)$ in an execution $\pi$ of a hyperLTS for $j, k \in [0, L]$, $j \neq k$, the atomic propositions that hold at $s(j)$ and $s(k)$ may be different, i.e., $p_s(j) \neq p_s(k)$ in a trace of $\pi$. Then, we say that $\pi$ has a trace satisfying an LTL$_{j}$ formula $\psi$ if and only if there exists a trace $\mu^*_s$ of $\pi$ such that $\mu^*_s \models \psi$.

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$_{j}$ 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 $\phi$ and the soft constraints are denoted by $\psi_1, \ldots, \psi_{N_S}$ 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 $\psi_i$ and $\psi_j$ 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 $\psi_n$ is more preferable than $\psi_m$ 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 $\Psi = \{(\psi_n, W_n)\mid n \in [1, N_S]\}$.

Problem 1: Given a hyperLTS $T^+$, a hard constraint $\phi$, a set of pairs of a soft constraint and its weight $\Psi$, and a positive integer $L$, find a trace $\mu^*_s$ of an execution $\pi$ with the length $L + 1$ of $T^+$ such that

- maximize: the sum of the weights of soft constraints that $\mu^*_s$ satisfies.
- subject to: $\mu^*_s \models \phi$.
- $\mu_s$ is a trace of an execution $\pi$ of $T^+$.

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 $T^+$ is given by $T^+ = (S^+, \delta^+, s^+_\text{init}, AP, L^+)$ with $|S^+| \geq 1$, its induced LTS is given by $T = (S, \delta, s_\text{init}, AP, L)$, where $S = \bigcup_{\pi \in S^+} (\{s^+_\pi\} \times L^+(s^+_\pi))$, $(s_1, s_2) \in \delta$ 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 $P_2 \in L^+(s^+_2)$, $s_\text{init} = (s^+_\text{init}, P)$ with $P \in L^+(s^+_\text{init})$, $L(s) = P$ 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$_{j}$ 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 $T^+$, where $s^+_\text{init} \in S^+ = \{s^+_1, s^+_2, \ldots, s^+_1\}$. Let $\pi = s(0)s(1)\ldots s(L)$ be a finite execution with the length $L + 1$ of $T^+$ and $s(0) = s^+_\text{init}$. We introduce $L + 1$ binary vectors $w(k) = [w_1(k), w_2(k), \ldots, w_{|S|^+}(k)]^T \in \{0, 1\}^{\sum_{s^+_\pi \in S^+} 1}$ for $k \in [0, L]$ to represent the $k$-th state $s(k)$ of $\pi$ as follows.

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

Since $s(0) = s^+_\text{init}$, we have $w_1(0) = 1$ if and only if $s^+_1 = s^+_\text{init}$. Denoted by $A \in \{0, 1\}^{|S|^+ \times |S|^+}$ is the transition matrix of $T^+$, where the $(i, j)$-th element $A_{ij}$ of $A$ is defined by

$$A_{ij} = \begin{cases} 1 & \text{if } (s^+_i, s^+_j) \in \delta^+, \\ 0 & \text{otherwise}. \end{cases}$$

This provides the following encoding.

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

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

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

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

Let $\mu^*_s = p_s(0)p_s(1)\ldots p_s(L)$ be a trace of the execution $\pi$ of $T^+$. We introduce $L + 1$ binary vectors $t(k) = [t_1(k), t_2(k), \ldots, t_{|S|^+}(k)]^T \in \{0, 1\}^{|S|^+}$ for $k \in [0, L]$ to represent the $k$-th set of atomic propositions $p_s(k)$ of $\mu^*_s$ as follows.

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

We introduce the labeling matrix of $T^+$, denoted by $V \in \{0, 1\}^{\sum_{s^+_\pi \in S^+} \times |S|^+}$, where the $(i, j)$-th element $V_{ij}$ of $V$ is defined by

$$V_{ij} = \begin{cases} 1 & \text{if } P_i \in L^+(s^+_j), \\ 0 & \text{otherwise}. \end{cases}$$
This provides the following encoding.
\[ t(k) \leq V w(k), \quad 1_{T^\dagger}^T t(k) = 1. \tag{2} \]
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 LTLₜ formula

For an LTLₜ formula \( \varphi \), we introduce the following binary variables \( z_\varphi(k) \) and \( y_\varphi(k) \) \( (k \in [0, L]) \) to encode the satisfaction of \( \varphi \) into a set of equations for \( \mu_x \) and \( \mu_z \), where \( \mu_x \) is a trace of an execution of \( T \), respectively.

\[ z_\varphi(k) = \begin{cases} 1 & \text{if } \mu_x(k...k) \models \varphi, \\ 0 & \text{otherwise}, \end{cases} \]
\[ y_\varphi(k) = \begin{cases} 1 & \text{if } \mu_z(k...k) \models \varphi, \\ 0 & \text{otherwise}. \end{cases} \]

Then, we encode atomic propositions for these traces.

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

\[ (v^{op})^T t(k) = z_\varphi(k). \tag{3} \]

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

\[ (u^{op})^T u'(k) = y_\varphi(k). \tag{4} \]

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 \( \varphi \) with (3) and (4), respectively.

### D. ILP problems

Based on the encodings for Problem 1, we convert this problem into the following ILP problem \( ILP_1 \) (resp. \( ILP_2 \)) for the approach using the hyperLTS (resp. the approach using the LTS).

**ILP1:**

\[
\text{maximize} \quad \sum_{n=1}^{N} W_n \cdot z_{\phi_n}(0) \\
\text{subject to} \quad (1), \ (2), \ ILP_1(\phi_1), \ldots, ILP_1(\phi_{N}), \text{ and } z_{\phi}(0) = 1.
\]

**ILP2:**

\[
\text{maximize} \quad \sum_{n=1}^{N} W_n \cdot y_{\phi_n}(0) \\
\text{subject to} \quad (1), \ ILP_2(\phi), \ldots, ILP_2(\phi_{N}), \text{ and } y_{\phi}(0) = 1.
\]

Note that both \( w(i), \ t(i), \) and \( w'(i) \) 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 \( T^+ \), the number of variables needed to encode an execution and a trace of the execution is \( |S^+| \cdot (L + 1) \) and \( |P| \cdot (L + 1) \), respectively. The sum of the number of variables needed to encode an execution and its trace for the hyperLTS is as follows.

\[ |S^+| \cdot (L + 1) + |P| \cdot (L + 1). \tag{5} \]

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

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

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

**Remark 1:** If we have

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

the number of variables for encoding 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 \( T^+ \) is as follows.

\[ O(|S^+| \cdot |S^+| + |P| \cdot |S^+| + |P|) = O(|S^+| \cdot (|S^+| + |P|)). \tag{8} \]

On the other hand, the order of the computation time to encode an execution of \( T \) is as follows.

\[ O(|S| \cdot |S| + |S|) = O(|S| \cdot |S|). \tag{9} \]

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 \( t \) and \( w \) for the approaches using the hyperLTS and the LTS, respectively.

Therefore, the orders of the computation time to encode (3) and (4) are \( O(|P|) \) and \( O(|S|) \), respectively. Note that we have

\[ |P| = \bigcup_{s \in S^+} |L^+(s)| \leq \sum_{s \in S^+} |L^+(s)| = |S|. \tag{10} \]

Thus, the computation time for the encoding of (3) is faster than that of (4). Since the encoding of all LTLₜ 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 $T^+ = (S^*, \delta^*, s^*_{\text{init}}, A^P, L^*)$ with $s^*_{\text{init}} = s^*_0$. 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 $S^*_i$ for $i \in \{1, 6\}$ where

$$
S^*_1 = \{s^*_0, s^*_1, \ldots, s^*_6\}, \quad S^*_2 = \{s^*_7, s^*_8, \ldots, s^*_10\}, \\
S^*_3 = \{s^*_11, s^*_12, \ldots, s^*_14\}, \quad S^*_4 = \{s^*_15, s^*_16, \ldots, s^*_19\}, \\
S^*_5 = \{s^*_20, s^*_21, \ldots, s^*_24\}, \quad S^*_6 = \{s^*_25, s^*_26, \ldots, s^*_30\}.
$$

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

$$
S^*_1 = \{s^*_2, s^*_3, s^*_4, s^*_5\}, \quad S^*_2 = \{s^*_6, s^*_7, s^*_8, s^*_9\}, \\
S^*_3 = \{s^*_10, s^*_11, s^*_12, s^*_13\}, \quad S^*_4 = \{s^*_14, s^*_15, s^*_16, s^*_17\}, \\
S^*_5 = \{s^*_18, s^*_19, s^*_20, s^*_21\}, \quad S^*_6 = \{s^*_22, s^*_23, s^*_24, s^*_25\}.
$$

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, $A^P$ is given by

$$
A^P = \{\text{Parts}_i, \text{Assemble}_i | i \in \{1, 12\}\}.
$$

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

$$
L^*(s^*) = \left\{
\begin{array}{ll}
\{\{\text{Parts}_{2i-1}, \text{Parts}_{2i}\}, \{\text{Parts}_{2i-1}, \text{Parts}_{2i}\}\} & \text{if } s^* \in S^*_i, \\
\{0, \text{Assemble}_{2i-1}, \text{Assemble}_{2i}\} & \text{if } s^* \in S^*_i, \\
\{0\} & \text{otherwise}.
\end{array}
\right.
$$

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

$$
\phi = \bigwedge_{i \in \{1, 12\}} \phi^*_i \wedge \bigwedge_{i \in \{1, 12\}} \phi^p_i,
$$

where $\phi^*_i = \mathcal{F}(\text{Assemble}_i)$, $\phi^p_i = (\neg \text{Assemble}_i) \mathcal{U}(\text{Parts}_i)$. The first term of (11) describes the robot has to assemble all products. For each $i \in \{1, 12\}$, $\phi^p_i$ 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

$$
\Psi = \{(\phi^p_{10}, 3), (\phi^p_{11}, 2), (\phi^p_{12}, 1), (\phi^p_{13}, 1), (\phi^p_{14}, 2), (\phi^p_{15}, 3)\},
$$

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

$$
\psi^i = (\neg \text{Assemble}_j) \mathcal{U}(\text{Assemble}_j).
$$

$\psi^i$ 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 plan pathing problems with L=30, 35, 40, 50, and 60 into Problem1 and convert them into ILP$_1$ and ILP$_2$.

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 $\pi$ for L=50 in the first case. From $\pi$, we have the following trace.

$$
t_0(0) = \{\text{Parts}_1, \text{Parts}_2\}, \quad t_0(2) = \{\text{Parts}_3, \text{Parts}_4\}, \\
t_0(4) = \{\text{Assemble}_1\}, \quad t_0(8) = \{\text{Assemble}_4\}, \\
t_0(14) = \{\text{Parts}_5, \text{Parts}_6\}, \quad t_0(15) = \{\text{Parts}_6\}, \\
t_0(17) = \{\text{Parts}_7, \text{Parts}_8\}, \quad t_0(19) = \{\text{Assemble}_7\}, \\
t_0(20) = \{\text{Assemble}_8\}, \quad t_0(26) = \{\text{Parts}_9, \text{Parts}_{10}\}, \\
t_0(28) = \{\text{Assemble}_9\}, \quad t_0(29) = \{\text{Assemble}_{10}\}, \\
t_0(35) = \{\text{Assemble}_{11}\}, \quad t_0(37) = \{\text{Assemble}_{12}\}, \\
t_0(38) = \{\text{Assemble}_{11}\}, \quad t_0(43) = \{\text{Assemble}_{15}\}, \\
t_0(44) = \{\text{Assemble}_{16}\}, \quad t_0(46) = \{\text{Assemble}_{22}\}, \\
t_0(50) = \{\text{Assemble}_{3}\},
$$

and $\emptyset$ are assigned to the other states in $\pi$. The trace of $\pi$ satisfies the hard constraint $\phi$ and the soft constraints other than $\psi^1_7$. To assemble all products with L=50, the robot assembles Product1 before it assembles Product7. Thus, $\psi^1_7$ is not satisfied. To satisfy $\psi^1_7$, the robot selects not to assemble Product3 on the first stay at the states in $S^*_2$. 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

<table>
<thead>
<tr>
<th>L</th>
<th># of variables</th>
<th>Encoding time (behavior)[s]</th>
<th>Encoding time (specification)[s]</th>
<th>Solving time[s]</th>
<th>Sum of weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10230</td>
<td>13175</td>
<td>199.34</td>
<td>546.34</td>
<td>18.50</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>35</td>
<td>11880</td>
<td>15300</td>
<td>217.25</td>
<td>601.48</td>
<td>19.83</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>40</td>
<td>13530</td>
<td>17425</td>
<td>252.04</td>
<td>687.58</td>
<td>22.83</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>50</td>
<td>16830</td>
<td>21675</td>
<td>310.70</td>
<td>868.46</td>
<td>28.14</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>60</td>
<td>20130</td>
<td>25925</td>
<td>376.56</td>
<td>1030.90</td>
<td>35.38</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>L</th>
<th># of variables</th>
<th>Encoding time (behavior)[s]</th>
<th>Encoding time (specification)[s]</th>
<th>Solving time[s]</th>
<th>Sum of weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>11067</td>
<td>9951</td>
<td>240.15</td>
<td>211.47</td>
<td>29.42</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>35</td>
<td>12852</td>
<td>11556</td>
<td>259.84</td>
<td>239.72</td>
<td>31.15</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>40</td>
<td>14637</td>
<td>13161</td>
<td>296.66</td>
<td>276.80</td>
<td>35.38</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>50</td>
<td>18207</td>
<td>16371</td>
<td>360.22</td>
<td>347.56</td>
<td>43.16</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
<tr>
<td>60</td>
<td>21777</td>
<td>19581</td>
<td>434.70</td>
<td>406.28</td>
<td>52.18</td>
</tr>
<tr>
<td></td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
<td>Hyper</td>
</tr>
</tbody>
</table>

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

REFERENCES


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 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.

VI. Conclusion