LTLC: Linear Temporal Logic for Control

YoungMin Kwon and Gul Agha

[1] Microsoft Corp.
ykwon4@cs.uiuc.edu
[2] Department of Comptuer Science
University of Illinois at Urbana Chamapaign
agha@cs.uiuc.edu

Abstract. Linear systems are one of the most commonly used models to represent physical systems. Yet, only few automated tools have been developed to check their behaviors over time. In this paper, we propose a linear temporal logic for specifying complex properties of discrete time linear systems. The proposed logic can also be used in a control system to generate control input in the process of model checking. Although, developing a full feedback control system is beyond the scope of this paper, authors believe that a feedback loop can be easily introduced by adopting the receding horizon scheme of predictive controllers. In this paper we explain the syntax, the semantics, a model checking algorithm, and an example application of our proposed logic.

1 Introduction

Linear systems have been widely used as mathematical models for physical systems because they can accurately represent the actual systems despite their simple structure. Thus, not surprisingly, many control systems are developed based on this simple mathematical model. In designing a control system, one of the fundamental questions about the system is the *controllability* of the system: whether we can drive the system from any state to any state [8]. Nowadays, with the popular use of versatile digital controllers, control systems can perform ever complex tasks and so become the requirements. In this challenging environment, one may want to know more than the traditional notion of controllability. For example, in a vehicle control system, we want to know, whether the vehicle can maintain certain speed even though we cannot keep accelerate it for longer than a duration to prevent overheating. The traditional controllability does not address this type of problem. Also, this seemingly simple problem has too many cases to be checked by hand: feasible set of state at each step depends on its past computational path – whether we accelerate or not at the current step affects the feasible set of state after the duration.

One obvious problem here is that we need a way to describe the complex requirements. Combinations of linear constraints can be a building block for the description: conjunctions of linear constraints define a convex region in the state space of the system and any arbitrary regions can be described by union or complements of them. However, these combinations of constraints can be too complex to generate or to modify by hand for nontrivial requirements. In this paper, we propose a logic, called *Linear Temporal*

Logic for Control (LTLC) on linear systems to describe the requirements in a highly abstract manner. LTLC uses logical and temporal operators to combine the constraints so that the complex path-dependent behaviors can be easily expressed. The usefulness of LTLC is not limited to checking the refined notion of controllability: it can also be used to compute a sequence of control input that can obtain control objectives.

Temporal logics like LTL, CTL, and CTL* are initially developed to specify behaviors of concurrent systems and later they are introduced to model checking [11, 6, 7]. Because their reasoning process is automated, model checking has been widely used in verifying complex hardware and software systems. However, the models of these logics are finite state machines whereas the model of LTLC is a linear system which has uncountably many states. Thus, in order to introduce these logics to linear systems we need different ways of expressing the states and different model checking algorithms. There has been approaches to address the problems of specifying and model checking in infinite state spaces. For example, Alur and Dill developed *timed automaton*, which is a finite state automaton with finite number of real valued clocks associated with the states, to model hybrid systems [2]. A decidability result for LTL model checking on controllable linear systems has been reported where a control system defined on a space of grid blocks bisimilar to the original linear system is built to divide the uncountable state space into finite partitions [12]. This result is extended to build a framework for designing controllers [13]. In iLTL, properties of Discrete Time Markov Chains (DTMC) are specified in the form of inequalities about expected rewards [10]. In iLTL the set of atomic propositions partitions the uncountable *probability mass function* (pmf) space into a finite number of equivalent classes. Although these approaches address the uncountable state space problem, none of these approaches address the question: "Given a system and a requirement, is there an initial state and a sequence of input that can drive the system to satisfy the specification?"

From the perspective of automatic control, *Model Predictive Control* (MPC) has similarity with our approach. MPC is an optimal control method minimizing a cost function of the error between the predicted output and the reference and of the energy to change the system state [4, 5]. One of the merits of MPC is that because it computes the predicted output it can follow non-constant references while satisfying other physical constraints. The key idea in the success of MPC is the use of finite input/output horizons where the input and output to and from the system become constant. These finite horizons enable us to express the output of the system in terms of a finite sequence of input. This conversion removes the dependency between input and output defined by the system dynamics equation during the optimization process. In LTLC model checking we adopt the key idea of MPC: adopt the input/output horizons and remove the input/output dependencies from the model checking process. However, unlike MPC where *always* enforced constraints are hard coded in the controller in the form of quadratic programming, LTLC provides a high level abstraction logic to generate complex sets of constraints. Note that this hard coded control objectives are difficult generate or modify because of the lack of the abstractions. Note that LTLC model checking can also be used to compute a sequence of control input to satisfy a complex control objective described in LTLC: a counter example of the negated control objective is a sequence of input that will satisfy the original goal.

2 Discrete Linear Time Invariant System Model

Our system model is a Discrete Linear Time Invariant System which can be represented by a seven-tuple $\mathcal{M} = (\ \mathcal{U},\ \mathcal{Y},\ \mathcal{X},\ \mathrm{A},\ \mathrm{B},\ \mathrm{C},\ \mathrm{D}\ )$, where $\mathcal{U} = \{u_1, \ldots, u_{nu}\}$ is a set of inputs, $\mathcal{Y} = \{y_1, \ldots, y_{ny}\}$ is a set of outputs, $\mathcal{X} = \{x_1, \ldots, x_{nx}\}$ is a set of states, and $\mathrm{A} \in \mathbb{R}^{nx \times nx}$, $\mathrm{B} \in \mathbb{R}^{nx \times nu}$, $\mathrm{C} \in \mathbb{R}^{ny \times nx}$, and $\mathrm{D} \in \mathbb{R}^{ny \times nu}$ are system matrices that describe the difference equations for the dynamics of the system. Our model describes a Multiple Input and Multiple Output (MIMO) system which has *nu* inputs and *ny* outputs.

In this paper, we overload the definitions of u_i, y_i, and x_i with the functions $u_i : \mathbb{N} \to \mathbb{R}$, $y_i : \mathbb{N} \to \mathbb{R}$, and $x_i : \mathbb{N} \to \mathbb{R}$ that map discrete time t to the value of input, output, and state at that time. We also define the following vector functions:

$$\begin{aligned}
\mathbf{u} : \mathbb{N} \to \mathbb{R}^{nu \times 1} &\text{ such that } \mathbf{u}(t)_i = u_i(t),\ \text{ for } i = 1, \ldots, nu,\\
\mathbf{y} : \mathbb{N} \to \mathbb{R}^{ny \times 1} &\text{ such that } \mathbf{y}(t)_i = y_i(t),\ \text{ for } i = 1, \ldots, ny,\\
\mathbf{x} : \mathbb{N} \to \mathbb{R}^{nx \times 1} &\text{ such that } \mathbf{x}(t)_i = x_i(t),\ \text{ for } i = 1, \ldots, nx,
\end{aligned}$$

where the subscript i of a vector is the i^{th} element of the vector.

The relations among input, output, and state functions are given by the following difference equations.

$$\begin{aligned}
\mathbf{x}(t+1) &= \mathrm{A} \cdot \mathbf{x}(t) + \mathrm{B} \cdot \mathbf{u}(t), \\
\mathbf{y}(t) &= \mathrm{C} \cdot \mathbf{x}(t) + \mathrm{D} \cdot \mathbf{u}(t).
\end{aligned} \tag{1}$$

Note that in the first difference equation, the next state $\mathbf{x}(t+1)$ is solely determined by the current state $\mathbf{x}(t)$ and the current input $\mathbf{u}(t)$. Thus, while inputs do not change, if two consecutive states remains the same, then the system is in a steady state from then on. That is, if $\mathbf{x}(t+1) = \mathbf{x}(t)$ and $\mathbf{u}(t+i) = \mathbf{u}(t)$ for $i \geq 0$ then $\mathbf{x}(t+j) = \mathbf{x}(t)$ for $j \geq 0$.

Given an input $\mathbf{u}$ and an initial state $\mathbf{x}(0)$, we can compute the state and the output of the system at time t as follows by recursively applying the equation (1).

$$\begin{aligned}
&\mathbf{x}(t) = \mathrm{A}^t \cdot \mathbf{x}(0) + \sum_{i=0}^{t-1} \mathrm{A}^{t-i-1} \cdot \mathrm{B} \cdot \mathbf{u}(i), \\
&\mathbf{y}(t) = \mathrm{C} \cdot \mathbf{x}(t) + \mathrm{D} \cdot \mathbf{u}(t).
\end{aligned} \tag{2}$$

3 Linear Temporal Logic for Control (LTLC)

In this section, we describe the syntax and the semantics of LTLC. LTLC has the same temporal and logical operators as *Linear Temporal Logic* (LTL). However, LTLC has different ways of describing atomic propositions than conventional LTL. A commonly used model for LTL is a Kripke structure [9] which is a finite state automaton with a set of atomic propositions associated with each state. In LTLC, with its uncountable state model, atomic propositions are given as a predicate function of states: equalities or inequalities about linear combinations of input, output, and state variables. With this form of atomic propositions, we can easily describe many useful properties of physical systems.

3.1 Syntax

The syntax of an LTLC formula ψ is as follows:

$$\begin{aligned}
\psi ::= {} & \mathtt{T} \mid \mathtt{F} \mid ap \\
& \neg\psi \mid \psi \vee \phi \mid \psi \wedge \phi \mid \psi \to \phi \mid \psi \leftrightarrow \phi \\
& \mathtt{X}\,\psi \mid \psi\,\mathrm{U}\,\phi \mid \psi\,\mathrm{R}\,\phi \mid \square\,\psi \mid \diamond\,\psi, \\
ap(t) ::= {} & c_1 \cdot v_1(texp_1) + \cdots + c_n \cdot v_n(texp_n) \bowtie d,
\end{aligned}$$

where ap is an atomic proposition, $texp_i$ is a polynomial of variable t, $c_1, \ldots, c_n$, and d are real numbers, $v \in \mathcal{U} \cup \mathcal{Y} \cup \mathcal{X}$ is one of input, output, or state variables, and $\bowtie$ is one of $\{<, \leq, >, \geq, =\}$.

As MPC enforces input and output *horizon constraints*, LTLC also enforces them. Note that they are not just constraints but an important control objective as well: drive the system to a steady state in finite time horizon. These constraints may restrict the scope of LTLC model checking but they play crucial roles in deriving the decidability result of Theorem 2. Also those computational paths pruned by these constraints are less interesting from an automatic control perspective: we are interested in those sequences of input that will drive the system to a steady state rather than arbitrary sequences of input. Let Hy be an output horizon when the system arrives a steady state and Hu be an input horizon ($Hu \leq Hy$) from which the inputs to the system do not change. This horizon constraint can be expressed as follows:

$$\bigwedge_{i=1}^{nx} x_i(Hy+1) = x_i(Hy) \wedge \bigwedge_{i=1}^{nu} u_i(Hu+j) = u_i(Hu) \text{ for } j > 0. \tag{3}$$

The *texp* of LTLC is a polynomial of time variable t. The use of *texp* enriches the expressiveness of LTLC such that some formula cannot be expressed otherwise. For example, in Pharmacokinetics, an instruction like take medicine at every three hours can be easily expressed in LTLC as: $\mathtt{always}\ (dose(3 \cdot t + 0) > 0\ \mathtt{and}\ dose(3 \cdot t + 1) = 0)\ \mathtt{and}\ dose(3 \cdot t + 2) = 0)$. However, improper use of *texp* can hamper the steady state constraints (3). Thus, we assume that all non-constant *texp* for state and output variables, $texp(t) \geq Hy$ for $t \geq Hy$ and that all non-constant *texp* for input variables, $texp(t) \geq Hu$ for $t \geq Hu$, where $texp(t)$ is the value of *texp* at time t.

3.2 Semantics

An LTLC formula has atomic propositions, logical connectives, $\neg$, $\vee$, $\wedge$, $\to$, and $\leftrightarrow$, and temporal connectives $\mathtt{X}$, $\mathtt{U}$, $\mathtt{R}$, $\square$, and $\diamond$. An atomic proposition of LTLC is a linear constraint on time-indexed variables (input, output, and state variables) with a comparator $\bowtie$. The meaning of an atomic proposition at any given time t is whether the linear constraint at time $texp(t)$ satisfies the usual meaning of $\bowtie$. Note that the value of state variables and the output variables can be rewritten in terms of an initial state and a sequence of inputs as can be seen in equation (2).

The meaning of logical operators $\neg$, $\vee$, and $\wedge$ are: $\neg\psi$ is true if and only if ψ is false, $\psi \vee \phi$ is true if and only if ψ or ϕ is true, and $\psi \wedge \phi$ is true if and only if ψ and ϕ are both

$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \mathtt{T}$		
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \not\models \mathtt{F}$		
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \sum_i c_i \cdot v_i(texp_i) \bowtie d$	$\Leftrightarrow$	$\sum_i c_i \cdot v_i(texp_i(t)) \bowtie d$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \neg\psi$	$\Leftrightarrow$	$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \not\models \psi$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi \wedge \phi$	$\Leftrightarrow$	$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi$ and $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \phi$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi \vee \phi$	$\Leftrightarrow$	$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi$ or $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \phi$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \mathtt{X}\,\psi$	$\Leftrightarrow$	$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t+1 \models \psi$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi\,\mathtt{U}\,\phi$	$\Leftrightarrow$	there is $j \geq 0$ such that $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t+j \models \phi$ and $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t+i \models \psi$ for $i = 0, \ldots, j-1$
$\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi\,\mathtt{R}\,\phi$	$\Leftrightarrow$	for all $i \geq 0$ if $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t+j \not\models \psi$ for $0 \leq j < i$ then $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t+i \models \phi$.

Fig. 1. Quintuple satisfaction relation $\models$.

true. The meaning of implies ($\rightarrow$) is $\psi \rightarrow \phi \;\Leftrightarrow\; \neg\psi \vee \phi$ and that of equivalent ($\leftrightarrow$) is $\psi \leftrightarrow \phi \;\Leftrightarrow\; \psi \rightarrow \phi \wedge \phi \rightarrow \psi$.

The meaning of temporal operators $\mathtt{X}$, $\mathtt{U}$, and $\mathtt{R}$ are: $\mathtt{X}\,\psi$ is true if and only if ψ is true at the next step, $\psi\,\mathtt{U}\,\phi$ is true if and only if ϕ eventually becomes true and before ϕ becomes true ψ is true, and $\psi\,\mathtt{R}\,\phi$ is true if and only if ϕ is true while ψ is false and if ψ becomes true then ϕ is true until that moment. The meaning of $\Box\,\psi$ is always ψ is true which is equivalent to $\mathtt{F}\ \mathtt{R}\,\psi$ and the meaning of $\Diamond\,\psi$ is eventually ψ becomes true which is equivalent to $\mathtt{T}\ \mathtt{U}\,\psi$.

Formally, the semantics of LTLC formula is defined by a binary satisfaction relation $\models\, \subset \mathcal{M} \times \psi$. In order to help explain the binary satisfaction relation $\models$, we overload the symbol and define a quintuple satisfaction relation $\models\, \subset \mathcal{M} \times (\mathbb{N} \rightarrow \mathbb{R}^{nu}) \times \mathbb{R}^{nx} \times \mathbb{N} \times \psi$ which is described in Figure 1. For simplicity we write $\mathcal{M} \models \psi$ for $(\mathcal{M}, \psi) \in\, \models$ and $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi$ for $(\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t, \psi) \in\, \models$.

The quintuple satisfaction relation is about a single path: whether a sequence of transitions from an initial state by a sequence of input satisfies the given LTLC formula. Using the definition of the quintuple satisfaction relation, the binary satisfaction relation $\models$ is defined as:

$$\mathcal{M} \models \psi \Leftrightarrow \mathcal{M}, \mathbf{u}, \mathbf{x}(0), 0 \models \psi \text{ for all } \mathbf{u}, \mathbf{x}(0).$$

The binary satisfaction relation is about all paths: whether the transitions from all initial states by all sequences of input satisfy the quintuple satisfaction relation.

In order to bring more insight into the syntax, the semantics, and usages of LTLC we explain the following example about drug administration. In Pharmacokinetics, drug concentrations in our body is often modeled as linear systems.

Example 1. Suppose that there is a patient who has disease in his lung. In order to cure the disease certain level of drug concentration (say, 5 mg/l) should be maintained in the lung for certain period of time (say, 3 hour). However, because this drug is toxic to liver, its concentration at the liver should not exceed certain level (say, 3 mg/l). Also, in order to increase absorption of the drug, it should be taken after dining, or say, every 4 hours. As a final condition, the drug should be cleared from the body eventually.

Let *dose* be the dose of medicine, *liver* is the concentration of the drug at the liver, and *lung* is the concentration of the drug at the lung.

The constraint that the drug should be taken at every 4 hours can be written as $\square\,(dose(4\cdot t+1)=0 \wedge dose(4\cdot t+2)=0 \wedge dose(4\cdot t+3)=0)$. In this formula the *always* operator $\square$ provides t from 0 to infinity. A similar but different formula is: $\square\,(dose(t)>0 \rightarrow (dose(t+1)=0 \wedge dose(t+2)=0 \wedge dose(t+3)=0))$. This formula can be read as, once he took the medicine he shouldn't take it again within 4 hours. Similarly, the drug concentration constraint in the liver can be written as $\square(liver(t)<3)$.

The goal, the condition about the drug concentration in the lung can be written as $\diamond\,(lung(t)>5 \wedge \mathtt{X}\,lung(t)>5 \wedge \mathtt{X}\,\mathtt{X}\,lung(t)>5)$. Note that the *eventually* operator $\diamond$ ensures that the condition should happen.

The last clearance condition can be written as: $\diamond\,\square\,(lung(t)=0 \wedge liver(t)=0)$. Note that the combined operators $\diamond\,\square$ specify properties at a steady state.

Finally, we can express the whole problem in LTLC as follows:

$$
\begin{aligned}
&\square\,(dose(4\cdot t+1)=0 \wedge dose(4\cdot t+2)=0 \wedge dose(4\cdot t+3)=0)\\
\wedge\;&\square\,(liver(t)<3)\\
\wedge\;&\diamond\,(lung(t)>5 \wedge \mathtt{X}\,lung(t)>5 \wedge \mathtt{X}\,\mathtt{X}\,lung(t)>5)\\
\wedge\;&\diamond\,\square\,(lung(t)=0 \wedge liver(t)=0).
\end{aligned}
$$

■

4 Model Checking

In this section, we describe an LTLC model checking algorithm. We first transform variables at different times into normal form, which is a fixed length coefficient vector. We then remove all temporal operators from the specification using the horizon constraints. Finally, we prove the decidability of LTLC model checking.

4.1 Converting timed variables to a normal form

The atomic propositions of LTLC are equality or inequality constraints about linear combinations of input, output, or state variables. These variables are related to others by the system dynamics equation (1). In this section we convert these timed variables into a normal form so that the dependencies among variables are eliminated during the model checking process. This is a standard technique in MPC to compute an optimal solution [4, 5]. In Section 3.1 we described the steady state constraint of LTLC. This constraint not only is a useful control objective but also makes LTLC model checking decidable. The constraint also plays a key role in defining the *normal form* explained below.

Let t_y be $\min(t, Hy)$ and let t_u and i_u be $\min(t, Hu)$ and $\min(i, Hu)$ respectively. If the steady state constraint (3) is satisfied then the system dynamics equation (2) can be rewritten as:

$$
\begin{aligned}
\mathbf{x}(t) &= \mathrm{A}^{t_y}\cdot\mathbf{x}(0) + \sum_{i=0}^{t_y-1}\mathrm{A}^{t_y-i-1}\cdot\mathrm{B}\cdot\mathbf{u}(i_u), \\
\mathbf{y}(t) &= \mathrm{C}\cdot\mathbf{x}(t) + \mathrm{D}\cdot\mathbf{u}(t_u).
\end{aligned}
\tag{4}
$$

For a constant c:	For an input variable $u_i(t)$:
$\mathbf{c}(c,t)_j = \begin{cases} c \text{ if } j = 1 \\ 0 \text{ otherwise} \end{cases}$	$\mathbf{c}(u,t)_j = \begin{cases} 1 \text{ if } j = 1 + nx + nu \cdot t_u + i \\ 0 \text{ otherwise} \end{cases}$
For a state variable $x_i(t)$: $\mathbf{c}(x,t)_1 = 0$ $\mathbf{c}(x,t)_{1+j} = (A^{t_y})_{ij}$ for $1 \le j \le nx$ $\mathbf{c}(x,t)_{1+nx+j\cdot nu+k} = \begin{cases} 0 & \text{if } j > t_y - 1 \\ \left(A^{t_y-j-1} \cdot B\right)_{ik} & \text{else if } j < Hu \\ \left(\Sigma_{j'=j}^{t_y-1} A^{j'} \cdot B\right)_{ik} & \text{else if } j = Hu \end{cases}$ for $0 \le j \le Hu, 1 \le k \le nu$	For an output variable $y_i(t)$: $\mathbf{c}(y,t)_1 = 0$ $\mathbf{c}(y,t)_{1+j} = (C \cdot A^{t_y})_{ij}$ for $1 \le j \le nx$ $\mathbf{c}(y,t)_{1+nx+j\cdot nu+k} = \begin{cases} \mathbf{c}'(y,t)_{j\cdot nu+k} + D_{ik} & \text{if } j = \min(Hu, t_y) \\ \mathbf{c}'(y,t)_{j\cdot nu+k} & \text{otherwise} \end{cases}$ for $0 \le j \le Hu, 1 \le k \le nu$, where $\mathbf{c}'(y,t)_{j\cdot nu+k} = \begin{cases} 0 & \text{if } j > t_y - 1 \\ \left(C \cdot A^{t_y-j-1} \cdot B\right)_{ik} & \text{else if } j < Hu \\ \left(\Sigma_{j'=j}^{t_y-1} C \cdot A^{j'} \cdot B\right)_{ik} & \text{else if } j = Hu \end{cases}$

Fig. 2. The conversion function **c**.

Note that in equation (4), $\mathbf{x}(t)$ or $\mathbf{y}(t)$ at any time t can be expressed in terms of $\mathbf{x}(0)$ and $\mathbf{u}(i)$ for $i = 0, \ldots, Hu$. Let $\mathbf{v}$ be a vector of these variables defined as:

$$\mathbf{v} = [1, x_1(0), \ldots, x_{nx}(0), u_1(0), \ldots, u_{nu}(0), \ldots, u_1(H_u), \ldots, u_{nu}(H_u)]^T.$$

Then, the normal form for a variable $z(t)$ is a coefficient vector, say $\mathbf{z}$, such that $z(t) = \mathbf{z} \cdot \mathbf{v}$, where z is one of input, output, or state variables. The conversion function $\mathbf{c}$: $(\mathcal{U} \cup \mathcal{Y} \cup \mathcal{X} \cup \mathbb{R}) \times \mathbb{N} \to \mathbb{R}^{1+nx+nu\cdot(Hu+1)}$ is defined in Figure 2. For simplicity, we overload the function $\mathbf{c} : (\mathcal{U} \cup \mathcal{Y} \cup \mathcal{X} \cup \mathbb{R}) \times \mathbb{N} \to \mathbb{R}^{1+nx+nu\cdot(Hu+1)}$ with $\mathbf{c} : AP \times \mathbb{N} \to AP$ as follows.

$$\begin{aligned} &\mathbf{c}(c_1 \cdot v_1(texp_1) + \cdots + c_n \cdot v_n(texp_n) \bowtie d, t) = \\ &\qquad (c_1 \cdot \mathbf{c}(v_1, texp_1(t)) + \cdots + c_n \cdot \mathbf{c}(v_n, texp_n(t)) - \mathbf{c}(d, 0)) \cdot \mathbf{v} \bowtie 0. \end{aligned}$$

With the normal form

$$\begin{aligned} &c_1 \cdot v_1(texp_1(t)) + \cdots + c_n \cdot v_n(texp_n(t)) \bowtie d \iff \\ &\qquad \mathbf{c}(c_1 \cdot v_1(texp_1) + \cdots + c_n \cdot v_n(texp_n) \bowtie d, t). \end{aligned}$$

The horizon constraint (3) can be written in LTLC formula as follows:

$$H : \bigwedge_{i=1}^{nx} (x_i(Hy+1) = x_i(Hy)) \wedge \bigwedge_{i=1}^{nu} \Box\, (u_i(Hu+t) = u_i(Hu))$$

Thus, given an LTLC formula ψ we implicitly mean $H \to \psi$.

4.2 Model checking as a feasibility checking

Before we explain the details of model checking algorithm, we first show an example that illustrates how to convert an LTLC model checking problem into a feasibility checking problem.

Example 2. Let a linear system $\mathcal{M}$ be $\left(\{u\}, \{y\}, \{x_1, x_2\}, \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, [1\ 1], \mathbf{0}\right)$, an atomic proposition $a(t)$ be $y(t) < 3$, horizon constraints be $Hu = 2$ and $Hy = 2$, and suppose that we want to find an initial state and a sequence of input such that $H \wedge \mathtt{X}\, a \wedge \mathtt{X}\,\mathtt{X}\, a$.

For this problem, we do LTLC model checking for the system $\mathcal{M}$ against a specification $\psi : H \rightarrow \neg(\mathtt{X}\, a \wedge \mathtt{X}\,\mathtt{X}\, a)$. Note that any counter example of $\mathcal{M} \models \psi$ satisfies the original goal. In practice, we search for $\mathbf{u}(t)$ and $\mathbf{x}(0)$ such that $\mathcal{M}, \mathbf{u}, \mathbf{x}(0), 0 \models \neg\psi$. That is,

$$\begin{aligned}
&\mathcal{M}, u, \mathbf{x}(0), 0 \models H \wedge \mathtt{X}\, a \wedge \mathtt{X}\,\mathtt{X}\, a \\
&\quad\Leftrightarrow \begin{cases} u(t+2) = u(2) \text{ for } t \geq 0 \wedge & \text{(input horizon constraint)} \\ x_1(2) = x_1(3) \wedge x_2(2) = x_2(3) \wedge & \text{(output horizon constraint)} \\ y(1) < 3 \wedge y(2) < 3 & (\mathtt{X}\, a \wedge \mathtt{X}\,\mathtt{X}\, a) \end{cases} \\
&\quad\Leftrightarrow \begin{array}{c} [0,4,3,4,3,2]\cdot\mathbf{v} = 0 \wedge \quad [0,6,2,6,3,1]\cdot\mathbf{v} = 0 \wedge \\ [-3,3,2,3,0,0]\cdot\mathbf{v} < 0 \wedge [-3,7,5,7,3,0]\cdot\mathbf{v} < 0 \end{array},
\end{aligned}$$

where $\mathbf{v}$ is $[1, x_1(0), x_2(0), u(0), u(1), u(2)]$. Thus,

$$\mathcal{M} \not\models \psi \Leftrightarrow \left\{\mathbf{v} : \begin{array}{c} [0,4,3,4,3,2]\cdot\mathbf{v} = 0 \wedge \quad [0,6,2,6,3,1]\cdot\mathbf{v} = 0 \wedge \\ [-3,3,2,3,0,0]\cdot\mathbf{v} < 0 \wedge [-3,7,5,7,3,0]\cdot\mathbf{v} < 0 \end{array}\right\} \neq \emptyset.$$

Note that the emptiness of $\mathbf{v}$ can be checked by linear programming and any feasible $\mathbf{v}$ is the counter example that we are seeking.

■

We now show how to transform an LTLC model checking problem into a feasibility checking problem and use it to prove the decidability of LTLC model checking. If the horizon constraint H is satisfied then the system arrives at a steady state from Hy step onward and the atomic propositions of the specification become constants. Otherwise, the system simply is not a model of the specification. If H is satisfied, then for $t \geq Hy$,

$$\begin{aligned}
\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \mathtt{X}\,\psi &\Leftrightarrow \mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi, \\
\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi\,\mathtt{U}\,\phi &\Leftrightarrow \mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \phi, \\
\mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \psi\,\mathtt{R}\,\phi &\Leftrightarrow \mathcal{M}, \mathbf{u}, \mathbf{x}(0), t \models \phi.
\end{aligned} \tag{5}$$

The use of normal form for timed variables and the fact that the system arrives at a steady state enable us to remove all the temporal operators from LTLC specifications. Figure 3 shows an algorithm to remove all temporal operators of an LTLC formula that is equivalent to the original formula if H is satisfied.

Theorem 1. $\mathcal{M} \models H \rightarrow \psi \;\Leftrightarrow\; \mathcal{M} \models H \rightarrow \mathtt{f}(\psi, 0)$.

```
f($\psi$,t) {
   if($\psi$ is T)     return T
   if($\psi$ is F)     return F
   if($\psi$ is an AP) return $\mathbf{c}(\psi, t)$
   if($\psi$ is $\neg\phi$)    return $\neg$f($\phi$,t)
   if($\psi$ is $\phi \vee \eta$)  return f($\phi$,t) $\vee$ f($\eta$,t)
   if($\psi$ is $\phi \wedge \eta$)  return f($\phi$,t) $\wedge$ f($\eta$,t)
   if($\psi$ is X $\phi$)
      if(t $\geq$ Hy)  return f($\phi$,t)
      else        return f($\phi$,t+1)
   if($\psi$ is $\phi$ U $\eta$)
      if(t $\geq$ Hy)  return f($\eta$,t)
      else        return f($\eta$,t) $\vee$ ( f($\phi$,t) $\wedge$ f($\psi$,t+1) )
   if($\psi$ is $\phi$ R $\eta$)
      if(t $\geq$ Hy)  return f($\eta$,t)
      else        return ( f($\phi$,t) $\wedge$ f($\eta$,t) ) $\vee$ ( f($\eta$,t) $\wedge$ f($\psi$,t+1) )
}
```

Fig. 3. Function f removes all temporal operators from an LTLC formula ψ.

Outline of proof: We prove the equivalence by induction on the structural tree of LTLC formula ψ. The induction base are T , F , and the atomic propositions which can easily proved. Induction steps on the logical connectives can be proved by the definitions of f and the quintuple satisfaction relation $\models$. Induction steps on the temporal connectives can be proved using the equivalence relations

$$\begin{aligned} \psi \,\mathrm{U}\, \phi &\equiv \phi \vee (\psi \wedge \mathtt{X}\,(\psi \,\mathrm{U}\, \phi)), \\ \psi \,\mathrm{R}\, \phi &\equiv (\psi \wedge \phi) \vee (\phi \wedge \mathtt{X}\,(\psi \,\mathrm{R}\, \phi)), \end{aligned} \tag{6}$$

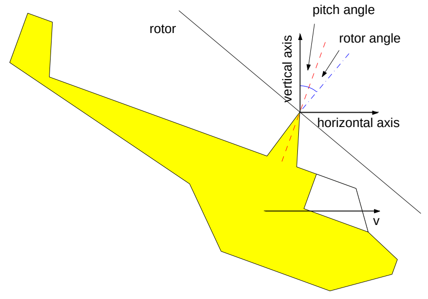
the steady state relation of equation (5), and the definition of f about the X formula. We divide the induction step on temporal operators in two cases: before time t reaches the steady state horizon, where we use the equivalence relation (6), and after the steady state, where we use the equation (5).

■

Theorem 2. *Model checking LTLC formulas $H \rightarrow \psi$ is decidable .*

Proof. Given an LTLC formula ψ with an implicit output horizon *Hy*, we can get a formula $\phi = \mathtt{f}(\psi, 0)$, which is equivalent to ψ, for the runs where H is true. Because ϕ does not have any temporal operators, we can transform it to Disjunctive Normal Form (DNF) whose satisfiability can be checked by checking the satisfiability of each conjunctive subformula. The subformula is a conjunction of linear equalities and inequalities in the normal form. Thus, the satisfiability of each subformula is equivalent to the feasibility of linear constraints which can be checked by linear programming. Because ϕ has only finite number of conjunctive subformulas and linear programming can be done in finite number of steps, LTLC model checking is decidable.

■



System dynamics

$$\begin{aligned}\mathbf{x}(t+1) &= \mathrm{A}\cdot\mathbf{x}(t) + \mathrm{B}\cdot\mathbf{u}(t)\\ \mathbf{y}(t) &= \mathrm{C}\cdot\mathbf{x}(t) + \mathrm{D}\cdot\mathbf{u}(t),\end{aligned}$$

where

$$A = \begin{bmatrix} 0.9680 & -0.0005 & -0.0010 \\ 0.0980 & 1 & 0 \\ -0.0888 & 0.9790 & 0.9981 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.6171 \\ 0.0311 \\ 0.9457 \end{bmatrix},\ C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},\ D = \mathbf{0}.$$

Fig. 4. A helicopter diagram and its discrete time dynamics.

Note that in LTLC model checking we do not make transitions of states explicitly. Instead, we transform the atomic propositions at different times into normal form. In model checking hybrid systems, one of the difficulties is the uncountably large state space. LTLC model checking addresses the difficulty in this framework. Thus, at any moment we can partition the state space into at most $|2^{AP}|$ equivalent classes which are not further distinguishable by AP.

Although the method described in this section is intuitive, when converted to a DNF, the number of conjunctive subformulas of an LTLC formula can grow exponentially in terms of the output horizon Hy. For example, the conjunctive subformulas of $\square\,(a \vee b)$ have $\bigwedge_{t=0}^{Hy}(a(t) \mid b(t))$, where $\mid$ is a choice operator. Thus, the number of conjunctive terms is 2^{Hy+1} and with large Hy model checking becomes practically impossible. Fortunately, however, there are many common terms in the conjunctive subformulas. If infeasibility is found in the common terms then we can skip checking all of the terms with the common infeasible terms. To leverage this computational benefit, we build a Büchi automaton [3] which can be thought as a generator of the conjunctive subterms. Each path of length Hy of the Büchi automaton is a conjunctive subformula. We can check the feasibility of common prefixes together and skip large number of redundant checks.

5 Experiment

In this section we illustrate how LTLC model checking can be used in controlling linear systems. The example system is a helicopter at near hover speed. Figure 4 shows a diagram of the example helicopter and its dynamics equations. The helicopter is composed of a body (fuselage) and a main rotor whose angle to the body is our control variable. The angle between the body and the horizontal plane is called *pitch* angle (nose down is positive). The speed of the helicopter (v in Figure 4) is defined at the center of mass. In this example we consider only the horizontal component of the velocity vector. The angle between the rotor plane and the direction of body is called *rotor angle*. The attitude (pitch angle pa, and pitch rate $pr = \dot{pa}$) and the speed (v) of a helicopter can be controlled by changing the rotor angle (dr).

```
#######################################
# System description
system:
    const pi = 3.141592;
    const rmin  = -pi*20/180, rmax = -rmin,
          rrmax =  pi*10/180;
    const A = [ 0.9608, -0.0005, -0.0010;
                0.0980,  1,       0;
               -0.0888,  0.9790,  0.9981 ],
          B = [ 0.6171;  0.0311;  0.9457 ];

    # System variables
    #   x:  pitch rate, pitch angle, speed
    #   dr: rotor angle
    #   v:  speed
    var x[3]: state,
        dr:   input,
        v:    output;

    # System dynamics equation
    x = A * x + B * dr;
    v = [0, 0, 1] * x;
```

```
#######################################
# Control objective description
specification:
    # when to bring the system to a steady state
    # and when to stop changing input
    output horizon: 25;
    input horizon: 24;

    # rotor angle constraints
    rp0(t): dr(t) >= rmin;
    rp1(t): dr(t) <= rmax;

    # rotor angular rate constraints
    rr0(t): dr(t+1) - dr(t) <= rrmax;
    rr1(t): dr(t+1) - dr(t) >= -rrmax;

    # initial state
    #   pitch rate and angle are both 0
    x0(t): x[0](t) = 0;
    x1(t): x[1](t) = 0;

    # initial and finial vehicle speed
    vi(t): v(t) = 2;    #initial speed
    vs(t): v(t+16) = 0; #stop at 1.6 sec
    vr(t): v(t) >= 5;   #finial speed

    # negated control objective
    ! (   [] ( rp0 /\ rp1 /\ rr0 /\ rr1 )
                  # physical constraints
      /\  x0 /\ x1  # initial state
      /\  vi        # initial speed
      /\  ([] vs \/ <> [] vr )
                  # either stop or speedup
      );
```

Fig. 5. LTLC specification for the experiment.

Our system model for the helicopter is $\mathcal{M} = (\mathcal{U}, \mathcal{Y}, \mathcal{X}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D})$, where $\mathcal{U} = \{dr\}$, $\mathcal{Y} = \{v\}$, $\mathcal{X} = \{pr, pa, v\}$, and the system dynamics equation is given in Figure 4. We obtain the discrete time dynamics by sampling a continuous time dynamics equations in [8] at a sampling rate of 10 samples per sec. We also consider physical constraints to the control variable (the rotor angle) to make the system more realistic: its maximum and minimum angles are +20 $^\circ$ and -20 $^\circ$ respectively, and its maximum and minimum angular rates are +100 $^\circ/sec$ and -100 $^\circ/sec$ respectively.

Figure 5 shows a model and specification description written in our LTLC checker [1]. It has two main components: a system description block that begins with `system:` tag and a specification block that begins with `specification:` tag. One can define scalar or matrix constants and type annotated variables in this block. Using the constants and the variables system dynamics equations are finally defined in this block. Note that the LHS of the dynamics equations are a state variable or an output variable.

The specification block begins with the implicit horizon constraints. The output horizon *Hy* and the input horizon *Hu* are first defined in this block as can be seen in Figure 5. Optional definitions of atomic propositions follow the horizon constraints. A definition of an atomic proposition has its name with a time variable and a linear constraint. A constraint is a comparison between linear combinations of input, output, and state variables. Also, each variable is associated with a time expression. In Figure 5, *rp0* and *rp1* describe the physical limits of the rotor angle, and *rr0* and *rr1* describe the limits of the rotor angular rate. The next two constraints *x0* and *x1* are about the state variable `x` (pitch rate and pitch angle). We use these equalities to specify an initial condition.

The last part of LTLC checker description is an LTLC formula using the previously defined atomic propositions. Usually, the topmost operator of the LTLC specification is the negation operator because we want to model check the negation of our control objective.

Now, suppose that the helicopter is flying at the speed of 2 m/sec and there is another vehicle approaching to it. In order to avoid collision we need to stop the helicopter within 1.6 sec or accelerate it to a speed faster than 5 m/sec within 2.5 sec. We want to know whether this control objective is achievable and if it is possible we want to know the input sequence also.

The subformula $x0 \wedge x1$ is about the initial state condition: the helicopter's initial pitch rate and pitch angle are both zero. Note that an LTLC formula without any temporal operator is about the initial step (at time 0). The subformula $\Box(rp0 \wedge rp1 \wedge rr0 \wedge rr1)$ means that the physical constraints on the rotor control are always imposed. The always operator $\Box$ ensures the binding of the time variable of inequalities and actual time during the process of model checking. Note the time expression *t+16* in *vs(t)*. Because the first time index of *vs(t)* is 16, the formula $\Box vs$ means that the helicopter stops from 1.6 sec onward. The formula $vi \wedge (\Box vs \vee \Diamond\Box vr)$ specifies that a vehicle initially flying at 2 m/sec speed stops within 1.6 sec or speeds up to a speed faster than 5 m/sec within 2.5 sec. Note how easily and intuitively the goal is expressed in LTLC. Even this simple disjunctive form of goal would be very difficult to write by hand.

The model checking result is:

```
result: F
state=   [ -0.000  0.000  2.000 ]^T
input[0]=[ -0.349 -0.349 -0.237 -0.063  0.112  0.286  0.349  0.349  0.175  0.287
            0.113  0.000  0.000 -0.157 -0.332 -0.157  0.000  0.000  0.000  0.000
            0.000  0.000  0.000  0.000  0.000                                    ]
```

In the model checking result '`result: F`' means that the helicopter is not the model of the negated specification. In other words, there is an initial state and a sequence of input that drive the system to meet the original control objective. As a counter example the model checker prints out the initial state and the sequence of input. Thus, by applying the input sequence in that order, the system will arrive at a steady state with all the control objectives satisfied.

Figure 6 shows the result of applying the computed input to the system from the computed initial state. In the second graph of Figure 6, the solid line is the control input, the dashed line is the resulting pitch rate, and the dot-dashed line is the resulting pitch angle of the helicopter. Notice the difference that the input sequence in the counter

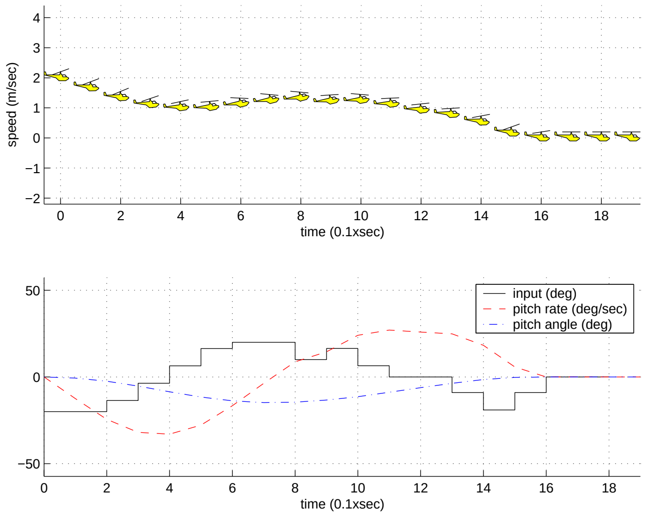


Fig. 6. Transitions of the model system driven by the computed input.

example is in radian whereas the graph is plotted in degree. From this graph we can see that the physical constraints on the rotor angle and its angular rate are always satisfied. The first graph of Figure 6 shows the helicopter's speed. This graph also shows the vehicle's pitch attitude and the rotor angle at the same time in order to give more insight into the dynamics of the system. Note that the vertical axis is the speed of the vehicle not the elevation. This graph shows how the vehicle comes to stop within 1.6 sec and the input also becomes constant from that moment.

6 Discussions

We developed a temporal logic called LTLC for specifying properties of linear systems and its model checking algorithm. LTLC model checking is decidable if we control the system to arrive at a steady state within a specified horizon. Although the implicit steady state constraints prevent LTLC model checking from using arbitrary input, many practical interest for the system require these constraints. LTLC can also be used to explicitly describe complex control objectives. A sequence of control input that can achieve the control objective can be computed in the process of model checking.

The use of *texp* in atomic propositions makes writing specification easy and intuitive. Also, *texp* extends the expressiveness of LTLC such that some properties cannot be expressed without it. Thus, *texp* can be regarded as a special temporal operator. However, on the other hand, its use can obscure the definition of state and requires a refinement in semantics. An interpretation of *texp* can be done in two layers: the first layer is a path determined by the choice of initial state and input; the second layer consists of the parallel compositions of reordered sequences of the path for each *texp* sampled at $texp(t)$.

As far as assuring system stability goes, the minimum bound for input horizon *Hu* and output horizon *Hy* are well known in predictive control literature [4]. In general a short horizon results in large input variation, which is not desirable from a control perspective, whereas long horizon slows down the model checking process. Also, the LTLC model checking for control described in this paper is for an ideal open loop control. If the linear system model is not accurate or if there are sensing noise or disturbances, the actual system output will deviate from the computed one and, if not corrected properly, the specification may be violated. A solution to these problems is to introduce a closed loop feedback control mechanism. The feedback control loop can be easily achieved by computing a new control input at every step with an updated state estimation as is commonly practiced in Receding Horizon Predictive Control scheme [4].

We believe, LTLC can be used as a high level abstraction tool that can hide the complexities of the underlying physical systems. We also believe that composing the abstractions to define higher level abstractions will be an important technique for handling the scalability problem in large systems.

References

1. LTLC Checker: `http://osl.cs.uiuc.edu/~ykwon4/cgi/LTLC.html`.
2. R. Alur and D.L. Dill. A theory of timed automata. In *Theoretical Computer Science*, volume 126, pages 183–235, 1994.
3. J.R. Büchi. On a decision method in restricted second order arthmetic. In *Proc. of the Int. Conf. on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1960.
4. D.W. Clarke, C. Mohtai, and P. Tuffs. Generalized predictive control. In *Automatica*, volume 23, pages 137–160, 1987.
5. D.W. Clarke and R. Scattolini. Constrained receding-horizon predictive control. In *IEE Proc. Part D*, volume 138, pages 347–354, 1991.
6. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logic of programs*. LNCS 131, Springer-Verlag, 1981.
7. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logics specification: A practical approach. In *Proc. 10th Int. ACM Symposium on Principles of Programming Languages*, pages 117–126, 1983.
8. Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison Wesley, 3rd edition, 1994.
9. G.E. Hughes and M.J. Creswell. *Introduction to Modal Logic*. Methuen, 1997.
10. YoungMin Kwon and Gul Agha. Linear inequality LTL (iLTL): A model checker for discrete time markov chains. In *International Conference on Formal Engineering Methods*, pages 194–208. LNCS 3308, 2004.
11. Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc of 12th ACM Symposium on Principles of Programming Languages*, pages 97–107, 1985.
12. Paulo Tabuada and George J. Papas. Model checking LTL over controllable linear systems is decidable. In *Hybrid Systems: Computation and Control:6th International Workshop*, volume LNCS 2623/2003, pages 498–513. Springer Berlin / Heidelberg, 2003.
13. Paulo Tabuada and George J. Papas. Linear time logic control of discrete-time linear systems. In *IEEE Transictions on Automatic Control*, volume 51, pages 1862–1877, 2006.