Abstraction-Refinement Based Optimal Control with Regular Objectives

Yoke Peng Leong, Student Member, IEEE, Pavithra Prabhakar, Member, IEEE,

Abstract

This paper presents an abstraction-refinement method to synthesize control inputs for a discrete-time piecewise linear system. The controlled system behavior satisfies a finite-word linear-time temporal objective while incurring minimal cost. An abstract finite state weighted transition system is constructed from finite partitions of the state and input spaces by solving optimization problems. A sequence of suboptimal controllers is obtained by considering a sequence of uniformly refined partitions. The abstract system satisfies the condition that the cost of the optimal control on the abstract system provides an upper bound on the cost of the optimal control for the original system. Furthermore, each suboptimal controller gives trajectories that have the cost upper bounded by the cost of the optimal control on the corresponding abstract system. In fact, the costs achieved by the sequence of suboptimal controllers converge to the optimal cost for the piecewise linear system. The tool OptCAR implements the abstraction-refinement algorithm. Examples illustrate the feasibility of this approach to synthesize automatically suboptimal controllers with improving optimal costs.

Index Terms

Optimal control, Hybrid systems, Formal methods, Abstraction-refinement.

I. INTRODUCTION

Formal synthesis is a paradigm for designing controllers automatically which are correct-by-construction, and thus, reduces the verification overhead. In this paradigm, a mathematical model of a system to be controlled and formal specifications of properties that are expected of the controlled system are given as inputs to compute a controller that ensures the controlled system satisfies the properties. For instance, given a model for the behavior of a robot, synthesize a plan that reaches a given part of the workspace while avoiding certain obstacles.

Since the work of [1] on automated synthesis, multiple directions are pursued including synthesizing finite state systems with respect to temporal logic objectives [2], [3] and controlling discrete event systems [4]. Early works in hybrid control systems focused on identifying subclasses of systems for which controller synthesis is decidable including timed automata [5], rectangular hybrid automata [6] and o-minimal automata [7], [8]. However, these

A version of this paper is submitted to the IEEE Transactions on Automatic Control. This version is updated with fixed typo found after the submission.

Y. P. Leong is with the Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA ypleong@caltech.edu
P. Prabhakar is with the Department of Computer Science, Kansas State University, Manhattan, KS 66506, USA pprabhakar@ksu.edu
classes of systems have limited continuous and discrete dynamics, and the synthesis problem becomes undecidable for a relatively simple class of hybrid systems [9].

For systems with complex dynamics, [10] introduced an abstraction based controller synthesis. Given a system, an abstract model, often a finite state system, is constructed such that a controller for the abstract model can be refined into a controller for the given system. The controller for the abstract model is constructed using results from automata theory, and it is then implemented in the given system. This method has been successfully applied in controller synthesis of switched systems [11], [12], and in robotic path planning [13], [14].

Often, in addition to designing a correct controller, an application may require optimality condition. For instance, a robot should reach a desired state with minimum battery. In this paper, we investigate an abstraction-refinement approach to synthesize optimal controller with regular properties that allow for specifications such as reaching a target region or traversing a sequence of regions. A regular property is specified as a (possibly) infinite set of finite traces interpreted as the allowed behaviors of the system, and generated by a finite state automaton. The foundation of our abstraction-refinement procedure and its correctness rely on defining appropriate preorders on the class of hybrid systems which preserve the optimal cost. This paper shows that the preordering defined satisfies the fact that if a system $H_2$ is higher up in the ordering than a system $H_1$, then the cost of the optimal controller for $H_1$ is at most that of $H_2$.

In our approach, first, an “abstraction” — a simplified finite state system — is constructed from partitions of the state and input spaces, and the edges of the system are annotated with weights which over-approximate the costs in the original system. Then, a two player game on the finite state system is solved to obtain a controller for the abstraction, and subsequently, a controller for the original dynamical system. This approach iteratively consider finer partitions of the state and input spaces, corresponding to grids of size $C/2^i$ for some constant $C$ and $i = 0, 1, 2, \ldots$. In fact, for discrete-time piecewise linear systems, if the cost function is continuous and the optimal control for the original system is robust with respect to the initial states, the cost of the sequence of controllers constructed converges to the optimal cost.

A. Related work and contributions

The main contribution of this paper is the optimality guarantee on the controllers synthesized — an important missing piece in most previous works that study optimal controller synthesis using formal approaches [15], [16]. A hierarchical optimal controller synthesis problem was studied [15], yet, no formal guarantees on the optimal cost are provided. Similarly, [16] considered optimal control synthesis by combining linear temporal logic, potential functions and model predictive control without formal guarantees on the optimal cost. On the other hand, the sequence of controllers constructed by our approach converges to the optimal cost for discrete time piecewise linear systems. Furthermore, for each suboptimal controller, the resulting trajectories have cost no greater than the optimal cost of the corresponding abstract system. Hence, when computational resources is limited, the best suboptimal controller found is guaranteed to generate trajectories with known bounded costs.

In addition, this technique is more general than classical finite horizon optimal control problems [17], [18] because the time horizon is not fixed a priori. Our approach focuses on finite horizon optimal control problems, but the
input sequence length is not a priori fixed because the regular property consists of finite traces whereby the length is variable. Apart from that, our method allows for a larger class of cost functions in comparison to previous works [19], [20], [21]. These works [19], [20], [21] used abstraction-based methods to find an optimal time controller that gives the shortest path which satisfies certain reachability conditions. In contrast, our method encodes transition cost in the abstraction scheme and thus, allowing for picking a path that is “shortest” with respect to a more general class of optimality conditions. Lastly, the technique presented in this paper does not place any prior restriction on the structure of the controllers [22], [23]. In [22], [23], trajectory based optimization is applied for synthesizing optimal control for discrete-time non-linear systems, however, it constrains the class of control strategies considered (to either finite paths or lassos).

The generality of our approach enables control engineers to synthesize controllers with more flexible structure and cost considerations. The method introduced in this paper applies to the general class of discrete-time hybrid systems. However, due to its’ generality, the computation burden could be high because the optimizations that compute the weights depend on the cost function and the dynamics. A prototype tool OptCAR that implements the abstraction refinement algorithm is presented (It will be made available for download when the paper is published). It is used to synthesize a (finite) sequence of controllers for a discrete-time linear system and a linear piecewise system with reachability objective.

A preliminary version of this work appeared in [24]. This extended version provides a more complete discussion of the technique including generalization of Theorem 6 with complete proof and an extra example to show that the resulting controller is similar to the linear quadratic regular for a linear system.

B. Paper Outline

The rest of this paper is organized as follows: Section II presents useful mathematical notations and definitions. Section III defines the weighted transition system and its relevant concepts, and Section IV explains the preorders for optimal control. The abstraction and refinement of a weighted transition system is developed in Section V. Section VI presents the problem formulation for optimal control of piecewise linear systems, the refinement procedure (OptCAR) and the cost analysis. The value iteration scheme for computing an optimal strategy for a finite transition system is described in Section VII. Section VIII presents the implementation of OptCAR using two examples. Lastly, Section IX summarizes the paper and states future directions of this work. Some proofs are provided in the Appendix at the end of this paper for ease of reading.

II. Notations

The sets of real numbers, non-negative real numbers, integers and non-negative integers are represented as \( \mathbb{R} \), \( \mathbb{R}_+ \), \( \mathbb{Z} \) and \( \mathbb{Z}_+ \), respectively. The set of integers \( \{0, \ldots, k\} \) is written as \([k]\) and a sequence \(x_0, \ldots, x_k\), denoted as \(\{x_i\}_{i \in [k]}\).

If \( M \in \mathbb{R}^{n \times m} \) is a matrix, \(\|M\|_\infty = \max_i \sum_{j=1}^m |M_{ij}|\). If \( z = (z_0, \ldots, z_k) \in \mathbb{R}^{k+1} \) is a vector, \(\|z\|_\infty = \max_i \left\{|z_i|\right\}_{i \in [k]}\).
An \( \epsilon \text{-ball around } x \) is defined as \( B_\epsilon(x) = \{ x' \in \mathbb{R}^n \mid |x - x'|_\infty \leq \epsilon \} \). Let \( S \subseteq \mathbb{R}^k \) be a \( k \)-dimensional subset. The function \( \text{Grid} \) splits \( S \) into rectangular sets with \( \epsilon \) width. That is, \( \text{Grid}(S, \epsilon) = \left\{ S' \mid \exists d_1, \ldots, d_k \in \mathbb{Z}, S' = S \cap \bigcap_{i=1}^{k} (d_i \epsilon, (d_i + 1) \epsilon) \right\} \).

Given a function, \( f : A \rightarrow B \), for any \( A \subseteq A \), \( f(A) = \{ f(a) \mid a \in A \} \). The domain of a function \( f \) is denoted as \( \text{dom}(f) \). Given an equivalence relation \( R \subseteq A \times A \) and an element \( a \in A \), \( [a]_R = \{ b \mid (a, b) \in R \} \) denotes the equivalence class of \( R \) containing \( a \).

### III. WEIGHTED TRANSITION SYSTEMS

This section defines a semantic model for discrete time hybrid systems with cost (i.e., weighted transition systems) and formalizes the optimal control problem.

**Definition 1.** A weighted transition system is defined as \( \mathcal{T} = (\mathcal{S}, \mathcal{S}^{\text{init}}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W}) \), where:

- \( \mathcal{S} \) is a set of states;
- \( \mathcal{S}^{\text{init}} \subseteq \mathcal{S} \) is a set of initial states;
- \( \mathcal{U} \) is a set of control inputs;
- \( \mathcal{P} \) is a set of propositions;
- \( \Delta \subseteq \mathcal{S} \times \mathcal{U} \times \mathcal{S} \) is a transition relation;
- \( \mathcal{L} : \mathcal{S} \rightarrow \mathcal{P} \) is a state labeling function, and
- \( \mathcal{W} : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow \mathbb{R}_+ \) is the transition cost function.

Note that an equivalent definition for the proposition set and the labeling function would be to let \( \mathcal{P}' \) be a set of propositions and define a labeling function that maps the states onto the power set of \( \mathcal{P}' \). The proposition set \( \mathcal{P} \) is related to \( \mathcal{P}' \) whereby \( \mathcal{P} = \{(p_0, \ldots, p_m) \mid p_i \in \mathcal{P}' \ \forall i \in [m]\} \). Current definition of proposition set is chosen for notational simplicity. In the sequel, a weighted transition system is referred as a transition system. For any \( s \in \mathcal{S} \), define the set \( \text{Enabled}(s) = \{ u \in \mathcal{U} \mid \exists s' \in \mathcal{S} \ s.t. (s, u, s') \in \Delta \} \) to represent all inputs that do not transitions the state \( s \) out of the predefined set \( \mathcal{S} \). A transition system is finite if \( \mathcal{S} \) and \( \mathcal{U} \) are finite. A finite state automaton (denoted \( \langle \mathcal{T}, \mathcal{P}_f \rangle \)) is a finite transition system \( \mathcal{T} \) along with a proposition \( \mathcal{P}_f \in \mathcal{P} \) which represents the final states.

For the rest of the section, fix the transition system \( \mathcal{T} = (\mathcal{S}, \mathcal{S}^{\text{init}}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W}) \).

a) **Paths and traces:** A path of the transition system \( \mathcal{T} \) is a sequence of states and inputs, \( \zeta = s_0 u_0 s_1 u_1 s_2 \ldots \), where \( s_0 \in \mathcal{S}^{\text{init}}, s_i \in \mathcal{S}, u_i \in \mathcal{U}, \) and \( (s_i, u_i, s_{i+1}) \in \Delta \). The set of all finite paths of \( \mathcal{T} \) is denoted \( \text{Paths}(\mathcal{T}) \). A trace of a transition system is the sequence of state labels of a path. The trace of \( \zeta \), denoted \( \text{Tr}(\zeta) \), is the sequence \( \mathcal{L}(s_0)\mathcal{L}(s_1)\ldots \).

b) **Properties:** This paper focuses on linear time properties over finite behaviors of systems. A property \( \Pi \) over a set of propositions \( \mathcal{P} \) is a set of finite sequences \( \pi = p_0 p_1 \ldots p_k \), where each \( p_i \in \mathcal{P} \). A property describes the desired behaviors of the system.

A property is regular if it consists of the traces of a finite state automaton \( \langle \mathcal{T}, \mathcal{P}_f \rangle \), that is, it is the set of all traces of paths of \( \mathcal{T} \) which start in an initial state and end in a state labelled by \( \mathcal{P}_f \). This paper considers regular property
that is specified by a finite state automaton \((T, P_f)\). Figure 1 shows an illustration. The properties expressed by popular logics such as finite words linear-time temporal logic (LTL) are regular, but their translation into the finite transition system representation can lead to an exponential blow up in the number of states with respect to the size of the formula [25].

Here, we consider regular properties that specify properties about finite behaviors as opposed to \(\omega\)-regular properties that specify properties about infinite behaviors. While our framework extends in a natural fashion to \(\omega\)-regular properties, the results related to the convergence discussed in Section VI do not extend to the \(\omega\)-regular case. In a separate future work, we intend to explore additional constraints for the synthesis problem with \(\omega\)-regular specifications that will ensure convergence of the control cost to the optimal value. Note that regular specifications already capture properties that are more general than finite horizon control, since a regular property can characterize behaviors involving unbounded length. For instance, consider reaching a target region without a priori bound on the number of steps required to reach the region.

c) Strategies: A strategy specifies the control inputs to a transition system at different time points. More specifically, a strategy \(\sigma\) for the transition system \(T\) is a partial function \(\sigma : \text{Paths}(T) \to \mathcal{U}\) such that for a path \(\zeta = s_0 u_0 s_1 \ldots u_{i-1} s_i\), \(\sigma(\zeta) \in \text{Enabled}(s_i)\). A path \(\zeta = s_0 u_0 s_1 u_1 s_2 \ldots\) is said to conform to a strategy \(\sigma\), if for all \(i\), \(\sigma(s_0 u_0 \ldots s_i) = u_i\).

A finite path \(\zeta = s_0 u_0 \ldots s_k\) maximally conforms to a strategy \(\sigma\), if \(\zeta\) conforms to \(\sigma\) and there is no extension \(\zeta' = s_0 u_0 \ldots s_k u_k s_{k+1}\) of \(\zeta\) which conforms to \(\sigma\). Let \(\text{Paths}^m_\sigma(T, s_0)\) denotes the maximally conforming finite paths of \(T\) with respect to \(\sigma\) starting at a state \(s_0\). Let \(\text{Str}(T)\) denote the set of all strategies which have no infinite
paths conforming to them. Note that the length of the paths which conform to a strategy in Str(\mathcal{T}) could still be variable.

To synthesize a strategy for \mathcal{T} from a state \textbf{s}_0 \in S^{ini} such that all maximal executions conforming to it reach a state in \mathcal{S}_f \subseteq \mathcal{S}, label the states in \mathcal{S}_f with a unique proposition. Then, let the property \Pi be the set of all traces corresponding to paths which start in \mathcal{S}_0^{ini} and end in \mathcal{S}_f, and do not visit \mathcal{S}_f in the middle.

**Definition 2.** A strategy \sigma for the transition system \mathcal{T} and an initial state \textbf{s}_0 \in \mathcal{S} is **winning** with respect to property \Pi over the propositions \mathcal{P}, if \sigma \in Str(\mathcal{T}) and Tr(Paths^\sigma_{\mathcal{T}}(\mathcal{T}, \textbf{s}_0)) \subseteq \Pi.

\textbf{d) Cost of strategies:} The cost of a path is the sum of the weights on the individual edges. Given a path \zeta = \textbf{s}_0 u_0 \textbf{s}_1 \ldots, define:

\[ W(\zeta) = \sum_j W(\textbf{s}_j, u_j, \textbf{s}_{j+1}). \]

Consequently, the following proposition holds.

**Proposition 1.** Given \zeta = \textbf{s}_0 u_0 \textbf{s}_1 \ldots \textbf{s}_k and \zeta' = \textbf{s}'_0 u'_0 s'_1 \ldots s'_k, if \[ W(\textbf{s}_j, u_j, \textbf{s}_{j+1}) \leq W(s'_j, u'_j, s'_{j+1}) \]
for all \textbf{j}, then \[ W(\zeta) \leq W(\zeta'). \]

This monotonicity property seems trivial, but plays an important role in the analysis later. In fact, results in this paper carry over for several other cost functions for paths such as average weight and maximum weight. Over infinite paths, average cost, maximum cost or discounted sum are more natural. Nonetheless, the analysis only relies on the fact that the cost of a path is monotonic with respect to the cost on the transitions. For simplicity, we fix one of the definitions.

The cost of a strategy \sigma of the transition system \mathcal{T} with respect to an initial state \textbf{s}_0 is defined as

\[ W(\mathcal{T}, \sigma, \textbf{s}_0) = \sup \{ W(\zeta) \mid \zeta \in Paths^\sigma_{\mathcal{T}}(\mathcal{T}, \textbf{s}_0) \}. \]

Accordingly, given a property \Pi over \mathcal{P}, the optimal cost of winning \mathcal{T} from an initial state \textbf{s}_0 with respect to a property \Pi is defined as

\[ W(\mathcal{T}, \textbf{s}_0, \Pi) = \inf \{ W(\mathcal{T}, \sigma, \textbf{s}_0) \mid \sigma \in Str(\mathcal{T}), Tr(Paths^\sigma_{\mathcal{T}}(\mathcal{T}, \textbf{s}_0)) \subseteq \Pi \}. \]

The cost is taken to be infinity if the minimization is over an empty set. Denote an optimal strategy that achieves the optimal cost as \sigma(\mathcal{T}, \textbf{s}_0, \Pi). Note that the optimal strategy may not be unique or exist.

\textbf{e) Optimal control problem:} Given the transition system \mathcal{T}, an initial state \textbf{s}_0 and a property \Pi, the optimal control problem is to compute an optimal winning strategy from \textbf{s}_0 with respect to \Pi, if it exists, and the optimal cost of winning.

\section*{IV. Preorders for Optimal Control}

In this section, a preorder on the class of transition systems is defined such that it preserves the optimal cost of winning. In other words, the optimal cost of winning in a system higher up in the ordering is an upper bound on
the optimal cost of winning in a system below it. For this, the definition of alternating simulations [26] is extended to include costs.

**Definition 3.** Given two transition systems $T_i = (S_i, S_i^{\text{init}}, U_i, \mathcal{P}, \Delta_i, L_i, W_i)$, for $i = 1, 2$, a simulation from $T_1$ to $T_2$ is a pair of relations $(\alpha, \beta)$, where $\alpha \subseteq S_1 \times S_2$ and $\beta \subseteq S_1 \times U_1 \times S_2 \times U_2$, such that:

1) $\forall (s_1, s_2) \in \alpha, L_1(s_1) = L_2(s_2)$.

2) $\forall s_1 \in S_1^{\text{init}}, \exists s_2 \in S_2^{\text{init}}$ such that $(s_1, s_2) \in \alpha$.

3) $\forall (s_1, s_2) \in \alpha$ and $u_2 \in \text{Enabled}(s_2), \exists u_1 \in \text{Enabled}(s_1)$ such that:

   a) $(s_1, u_1, s_2, u_2) \in \beta$.

   b) $\forall (s_1, u_1, s'_2) \in \Delta_1, \exists (s_2, u_2, s'_2) \in \Delta_2$ such that $(s'_1, s'_2) \in \alpha$ and $W_1(s_1, u_1, s'_2) \leq W_2(s_2, u_2, s'_2)$.

Let $T_1 \preceq_{(\alpha, \beta)} T_2$ denote that $(\alpha, \beta)$ is a simulation from $T_1$ to $T_2$. If there exists some $(\alpha, \beta)$ such that $T_1 \preceq_{(\alpha, \beta)} T_2$, then $T_2$ simulates $T_1$, and it is denoted as $T_1 \preceq T_2$.

**Theorem 2.** $\preceq$ is a preorder on the class of transition systems over a set of propositions $\mathcal{P}$.

**Proof:** Define $(\alpha, \beta)$ to be identity relations on the state and input spaces, then $T \preceq_{(\alpha, \beta)} T$, and hence $\preceq$ is reflexive. To show that $\preceq$ is transitive, suppose $T_1 \preceq_{(\alpha_1, \beta_1)} T_2$ and $T_2 \preceq_{(\alpha_2, \beta_2)} T_3$. Define $\alpha$ such that $(s_1, s_3) \in \alpha$ if $(s_1, s_2) \in \alpha_1$ and $(s_2, s_3) \in \alpha_2$ for some $s_2$, and define $\beta$ such that $(s_1, s_3) \in \beta$ if $(s_1, u_1, s_2, u_2) \in \beta_1$ and $(s_2, u_2, s_3, u_3) \in \beta_2$ for some $(s_2, u_2)$. Then, $T_1 \preceq_{(\alpha, \beta)} T_3$.

The next result shows that $\preceq$ is an ordering on the transition systems which “preserves” optimal control.

**Theorem 3.** Given two transition systems $T_i = (S_i, S_i^{\text{init}}, U_i, \mathcal{P}, \Delta_i, L_i, W_i)$ for $i = 1, 2$, let $\Pi$ be a property over a set of propositions $\mathcal{P}$. $T_1 \preceq_{(\alpha, \beta)} T_2$ and $(s_0, s'_0) \in \alpha$ for $s_0 \in S_0^{\text{init}}$ and $s'_0 \in S_2^{\text{init}}$. If there exists a winning strategy $\sigma_2$ for $T_2$ from $s'_0$ with respect to $\Pi$, then there exists a winning strategy $\sigma_1$ for $T_1$ from $s_0$ with respect to $\Pi$ such that $W_1(T_1, \sigma_1, s_0) \leq W_2(T_2, \sigma_2, s'_0)$. Hence, $W_1(T_1, s_0, \Pi) \leq W_2(T_2, s'_0, \Pi)$.

**Proof:** Let $\sigma_2$ be a strategy for $T_2$ and $s'_0$. In addition, define a partial mapping $G : \text{Paths}(T_1) \rightarrow \text{Paths}(T_2)$ such that the domain of $G$ is the set of all paths from $s_0$ that conform to $\sigma_1$, and for any path $\zeta_1$ in the domain of $G$, $L_1(\zeta_1) = L_2(G(\zeta_1))$, and $W_1(\zeta_1) \leq W_2(G(\zeta_1))$. This construction ensures that if $\sigma_2$ is winning from $s'_0$ with respect to $\Pi$, then so is $\sigma_1$ from $s_0$ and $W_1(T_1, \sigma_1, s_0) \leq W_2(T_2, \sigma_2, s'_0)$. We also ensure that if $G(\zeta_1) = \zeta_2$, then $(s_k, s'_k) \in \alpha$, where $s_k$ and $s'_k$ are the end states of $\zeta_1$ and $\zeta_2$, respectively. Further, for any $\zeta_1$ in the domain of $G$, $\zeta_1$ is a maximal path conforming to $\sigma_1$ if and only if $\zeta_2$ is a maximal path conforming to $\sigma_2$.

Next, define $\sigma_1$ and $G$ by induction on the length of words in their domain. Set $G(s_0) = s'_0$. Suppose $\sigma_1$ for paths of length $k - 1$ and $G$ for paths of length $k$, are defined such that the invariant holds. Let $\zeta_1 = s_0 u_0 s_1 \ldots s_k$ conform to $\sigma_1$. Then, $G(\zeta_1)$ is defined. Let $G(\zeta_1) = s'_0 u'_0 s'_1 \ldots s'_k$ and $(s_k, s'_k) \in \alpha$. If $G(\zeta_1)$ is a maximal path conforming to $\sigma_2$, then $\sigma_1(\zeta_1)$ is not defined (i.e., $\zeta_1$ is not in the domain of $\sigma_1$). Otherwise $\sigma_2(G(\zeta_1)) = u'_k$. Then, from the second condition of simulation, there exists $u_k$ such that $(s_k, u_k, s'_k, u'_k) \in \beta$. Choose $\sigma_1(\zeta_1) = u_k$. For any $\zeta_2 = s_0 u_0 s_1 \ldots s_{k+1}$, define $G(\zeta_2) = s'_0 u'_0 s'_1 \ldots s'_{k+1}$ such that $(s_{k+1}, s'_{k+1}) \in \alpha$ and $W_1(s_k, u_k, s_{k+1}) \leq W_2(s'_k, u'_k, s'_{k+1})$. It can be verified that the construction satisfies the inductive invariant.
V. ABSTRACTION/REFINEMENT

In this section, the abstraction refinement procedure for constructing finite state systems which simulate a given transition system is presented. The state and input spaces are divided into finite number of parts, and they are used as symbolic states and inputs, respectively, in the abstract transition system. Henceforth, fix a transition system $T = (S, S_{init}, U, P, \Delta, L, W)$.

Definition 5. A transition system $T$ is a complete transition system if for all $s \in S$, $U = Enabled(s)$.

A. Abstraction

An abstraction function constructs an abstract transition system $Abs(T, \equiv_S, \equiv_U)$ given the transition system $T$, and two equivalence relations $\equiv_S$ and $\equiv_U$ on the state-space $S$ and the input-space $U$, respectively. To ensure a well defined abstract transition system, $\equiv_S$ on $S$ needs to respect both the set of labels $L$ and the set of initial states $S_{init}$. In other words, the labels are the same for all equivalent states, and the initial states in the set $S_{init}$ are not equivalent to any states outside of the set $S_{init}$. More formally, an equivalence relation $\equiv_S$ on $S$ respects $L$, if for all $(s_1, s_2) \in \equiv_S$, $L(s_1) = L(s_2)$. Furthermore, an equivalence relation $\equiv_S$ on $S$ respects $S_{init}$, if for all $(s_1, s_2) \in \equiv_S$ where $s_1 \in S_{init}$, $s_2 \in S_{init}$.

Definition 5. Let $\equiv_S \subseteq S \times S$ and $\equiv_U \subseteq U \times U$ be two equivalence relations of finite index such that $\equiv_S$ respects the labeling function $L$ and the initial states $S_{init}$. $Abs(T, \equiv_S, \equiv_U) = (S', S_{init}', U', P, \Delta', L', W')$, where:

- $S' = \{[s]_{\equiv_S} \mid s \in S\}$ is the equivalence classes of $\equiv_S$.
- $S_{init}' = \{[s]_{\equiv_S} \mid s \in S_{init}\} \subseteq S'$.
- $U' = \{[u]_{\equiv_U} \mid u \in U\}$ is the equivalence classes of $\equiv_U$.
- $\Delta' = \{(S_1, U, S_2) \mid \exists s \in S_1, s' \in S_2, u \in U, s.t. (s, u, s') \in \Delta\}$.
- For $s \in S'$, $L'(s) = L(s)$ for any $s \in S$.
- For $(S_1, U, S_2) \in \Delta'$, $W'(S_1, U, S_2) = \sup\{W(s_1, u, s_2) \mid s_1 \in S_1, s_2 \in S_2, u \in U, (s_1, u, s_2) \in \Delta\}$.

Call $T$ the concrete system and $Abs(T, \equiv_S, \equiv_U)$ the abstract system. The next proposition states that the abstract system simulates the concrete system.

Proposition 4. If $T$ is a complete transition system, $T \preceq Abs(T, \equiv_S, \equiv_U)$.

Proof: Define $\alpha = \{(s, [s]_{\equiv_S}) \mid s \in S\}$, and $\beta = \{(s, u, [s]_{\equiv_S}, [u]_{\equiv_U}) \mid s \in S$ and $u \in U\}$. Then, properties in Definition 3 are satisfied.

Consider $Abs(T, \equiv_S, \equiv_U) = (S', S_{init}', U', P, \Delta', L', W')$ as in Definition 5. Define $(s, [s]_{\equiv_S}) \in \alpha$ for $s \in S$, and $(s, u, [s]_{\equiv_S}, [u]_{\equiv_U}) \in \beta$ for $s \in S$ and $u \in U$.

The first property in Definition 3 is satisfied by construction because for all $S \in S'$, $L'(s) = L(s)$ for any $s \in S$. The second property also holds by construction of $S_{init}'$ where $\forall s \in S_{init}$, there exists a $[s]_{\equiv_S} \in S_{init}'$ such that $(s, [s]_{\equiv_S}) \in \alpha$. 

To verify the third property, consider any \((s_1, S_1) \in \alpha\) and \(U \in \text{Enabled}(S_1)\). Because \(U \in \mathcal{U}'\), there exists a \(u \in \mathcal{U}\) where \([u]_{\equiv_U} = U\). Given that \(\text{Enabled}(s_1) = U\) for a complete transition system, \(u \in \text{Enabled}(s_1)\). By definition of \(\beta\), \((s, u, S_1, U) \in \beta\). Furthermore, for \((s_1, u, s_2) \in \Delta\), there exists \(S_1 \in \mathcal{S}'\) such that \((s_1, S_1) \in \equiv_S\), \(S_2 \in \mathcal{S}'\) such that \((s_2, S_2) \in \equiv_S\), and \(U \in \mathcal{U}'\) such that \((u, U) \in \equiv_U\). Furthermore, by construction, \((S_1, U, S_2) \in \Delta'\) if \((s_1, u, s_2) \in \Delta\). Thus, there exists a \((S_1, U, S_2) \in \Delta'\) where \((s_2, S_2) \in \alpha\). Lastly, \(W'(S_1, U, S_2) \geq W(s_1, u, s_2)\) because \(W'\) is the maximum over all \(s_1 \in S_1, u \in U, s_2 \in S_2\) of \(W(s_1, u, s_2)\).

\[\Box\]

B. Refinement

We can construct a sequence of abstract systems which are closer to the original system than their predecessors in the sequence, by choosing finer equivalence relations on the state and input spaces.

**Definition 6.** Let \(\mathcal{T}_1\) and \(\mathcal{T}_3\) be transition systems such that \(\mathcal{T}_1 \preceq \mathcal{T}_3\). A transition system \(\mathcal{T}_2\) is said to be a refinement of \(\mathcal{T}_3\) with respect to \(\mathcal{T}_1\), if \(\mathcal{T}_1 \preceq \mathcal{T}_2 \preceq \mathcal{T}_3\).

**Proposition 5.** Let \(\equiv_S, \equiv_S' \subseteq \mathcal{S} \times \mathcal{S}\) and \(\equiv_U, \equiv_U' \subseteq \mathcal{U} \times \mathcal{U}\) be equivalence relations of finite index such that \(\equiv_S' \subseteq \equiv_S\) and \(\equiv_U' \subseteq \equiv_U\). Then, \(\text{Abs}(\mathcal{T}, \equiv_S, \equiv_U)\) is a refinement of \(\text{Abs}(\mathcal{T}, \equiv_S', \equiv_U')\) with respect to \(\mathcal{T}\).

**Proof:** First, \(\mathcal{T} \preceq \text{Abs}(\mathcal{T}, \equiv_S', \equiv_U')\) follows from Proposition 4. Define \(\alpha = \{([s]_{\equiv_S'}, [s]_{\equiv_S}) | s \in \mathcal{S}\}\) and \(\beta = \{([s]_{\equiv_S}, [u]_{\equiv_U}, [s]_{\equiv_S}, [u]_{\equiv_U}) | s \in \mathcal{S} \text{ and } u \in \mathcal{U}\}\). Then, properties in Definition 3 are satisfied for \(\text{Abs}(\mathcal{T}, \equiv_S', \equiv_U') \preceq \text{Abs}(\mathcal{T}, \equiv_S, \equiv_U, \mathcal{T})\), and thus, \(\mathcal{T} \preceq \text{Abs}(\mathcal{T}, \equiv_S', \equiv_U') \preceq \text{Abs}(\mathcal{T}, \equiv_S, \equiv_U)\).

\[\Box\]

VI. OPTIMAL CONTROL OF PIECEWISE LINEAR SYSTEMS

This section considers an optimal control problem for discrete-time piecewise linear systems. The abstraction refinement approach is applied to construct a series of controllers with improving suboptimal costs that converge to the optimal cost under the existence of a robust optimal control.

A. Problem Formulation

A discrete-time piecewise linear system is a tuple \((\mathcal{X}, \mathcal{X}^{\text{init}}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i \in [m]}, \mathcal{L}_d, \mathcal{J})\), where the state-space \(\mathcal{X} \subseteq \mathbb{R}^n\) and the input-space \(\mathcal{U} \subseteq \mathbb{R}^p\) are compact sets, \(\mathcal{X}^{\text{init}} \subseteq \mathcal{X}\) is the set of initial states, \(\mathcal{P}\) is a finite set of propositions, \(A_i \in \mathbb{R}^{n \times n}, B_i \in \mathbb{R}^{n \times p}\) and \(P_i\) is a polyhedral set, such that \(\{P_i\}_{i \in [m]}\) is a polyhedral partition of \(\mathcal{X}\), \(\mathcal{L}_d : [m] \rightarrow \mathcal{P}\) is a labeling function and \(\mathcal{J} : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}_+\) is a continuous cost function. Note that \(A_i\) and \(B_i\) can be the same for different \(i\). We associate a unique label to each region \(P_i\). We could have assigned different labels to different regions in some polyhedral partition of \(P_i\); we do not lose expressiveness here, since, the latter can be transformed to the former problem by considering a finer partition whose regions are the regions partitioning each \(P_i\) according to the label.

Given an initial state \(x_0 \in \mathcal{X}^{\text{init}}\) and a sequence of control inputs \(u = \{u_t\}_{t \in [k]}\), where \(u_t \in \mathcal{U}\), \(\phi(x_0, u) = \{x_t\}_{t \in [k+1]}\) is the sequence of states visited under the control \(u\), where \(x_{t+1} = A_i x_t + B_i u_t\), and \((A_i, B_i) = (A_i, B_i)\) if \(x_t \in P_i\). The cost of the sequence \(\phi(x_0, u), \mathcal{J}(\phi(x_0, u))\), is given by \(\sum_{t \in [k]} \mathcal{J}(x_{t+1}, u_t)\). We define the partition
sequence of \( \{x_t\}_{t \in [k+1]} \), denoted \( PS(\{x_t\}_{t \in [k+1]}) \), to be the sequence of partitions visited by the states, namely, \( P_1, \ldots, P_{k+1} \) such that \( x_t \in P_i \) for all \( t \in [k + 1] \).

**Problem 1** (Optimal control problem).

Given an \( n \)-dimensional discrete-time piecewise linear system \( D = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i \in [m]}, L_d, J) \), a state \( x_0^* \in \mathcal{X}^{init} \) and a regular property \( \Pi \) over \( \mathcal{P} \), find a sequence of control inputs \( u^* \) for which \( L_d(\phi(x_0^*, u^*)) \in \Pi \) and \( J(\phi(x_0^*, u^*)) \) is minimized.

**Remark 1.** Although the property \( \Pi \) is over finite sequences, it could potentially contain finite sequences of unbounded lengths (i.e. no fixed upper bound on the sequence length). Hence, the Problem 1 is not the same as a classical finite horizon problem, because the optimal control sequence length is not fixed a priori.

**B. Solution**

A discrete-time piecewise linear system \( D = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i \in [m]}, L_d, J) \) can be represented as a weighted transition system, \( \mathcal{T}_D = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \Delta, L, W) \) where \( \Delta = \{(x, u, x') \in \mathcal{X} \times \mathcal{U} \times \mathcal{X} | x' = Ax + Bu \) where \( (A, B) = (A_i, B_i) \) for \( x \in P_i \) \}, \( L(x) = L_d(i) \) where \( x \in P_i \), and \( W(x, u, x') = J(x', u) \).

Consequently, Problem 1 is equivalent to the following problem:

**Problem 2** (Optimal strategy problem).

Given a weighted transition system \( \mathcal{T}_D = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \Delta, L, W) \), a state \( x_0^* \in \mathcal{X}^{init} \) and a regular property \( \Pi \) over \( \mathcal{P} \), find an optimal winning strategy \( \sigma(\mathcal{T}_D, x_0^*, \Pi) \) for which the optimal cost of winning \( \mathcal{T}_D \) with respect to \( \Pi, W(\mathcal{T}_D, x_0^*, \Pi) \), is achieved.

Note that since \( \mathcal{T}_D \) is input deterministic, \( \sigma(\mathcal{T}_D, x_0^*, \Pi) \) will correspond to a unique path starting from \( x_0^* \).

In general, solving Problem 2 is difficult, since, \( \mathcal{T}_D \) is a infinite state system; hence, we focus on synthesizing suboptimal strategies using Algorithm 1. As an overview, Algorithm 1 first partitions the state space into grids of a particular size, and constructs an abstract system for the system \( \mathcal{T}_D \). Then, it computes the optimal cost \( J \) and strategy of the abstract system through a two-player game. A suboptimal strategy for \( \mathcal{T}_D \) can then be extracted.

---

**Algorithm 1 OptCAR (Abstraction Refinement Procedure)**

**Require:** System \( D \), Property \( \Pi \) as a finite state automaton, initial state \( x_0^* \), rational number \( 0 < \epsilon_0 \)

Set \( \epsilon := \epsilon_0 \)

while true do

\( \hat{\mathcal{T}}, \hat{x}_0 := ConsAbs(D, \epsilon) \)

\( J, \hat{\sigma} := SolveFiniteGame(\hat{\mathcal{T}}, \hat{x}_0, \Pi) \)

\( \sigma_D := ExtractController(\hat{\sigma}, \hat{\mathcal{T}}, D) \)

Output \( \sigma_D \) and \( J \)

\( \epsilon := \frac{\epsilon}{2} \)

end while
from the strategy of the abstract system with the cost upper bounded by $J$. If the upper bound $J$ is not zero, refine the state space partitions using smaller grids, and repeat the whole process to reduce the cost $J$. As a result, this algorithm outputs a sequence of suboptimal strategies, whose costs converge to that of the optimal cost.

More precisely, in each iteration, Algorithm 1 performs the following sequence of steps. First, it constructs a finite state abstraction $\hat{T}$ of $D$ using the function $\text{ConsAbs}(D, \epsilon)$. $\text{ConsAbs}(D, \epsilon)$ outputs $\text{Abs}(\overline{T_D}, \equiv_X, \equiv_U)$, where $\equiv_X$ and $\equiv_U$ are equivalence relations whose equivalences classes are the elements of $\text{Grid}(X, \epsilon)$ and $\text{Grid}(U, \epsilon)$, respectively. Define the initial abstract state as $\hat{x}_0 := [x_0^\epsilon]_{\equiv_X}$. This step solves $|S|^2|U|$ optimizations where $|S|$ is the number of states in $\hat{T}$ and $|U|$ is the number of control inputs in $\hat{T}$. These optimizations can be computed in parallel. Next, $\text{SolveFiniteGame}(\hat{T}, \hat{x}_0, \Pi)$ computes the optimal cost of winning $J = W(\hat{T}, \hat{x}_0, \Pi)$ with respect to $\Pi$ in the finite state transition system $\hat{T}$ and the corresponding strategy $\hat{\sigma} = \sigma(\hat{T}, \hat{x}_0, \Pi)$ for $\hat{T}$ through a two-player game (see Algorithm 2 of Section VII for more details).

Finally, $\text{ExtractController}(\hat{\sigma}, \hat{T}, D)$ outputs a suboptimal strategy/controller $\sigma_D$ whose cost is bounded by the optimal cost $J$ for the abstract system. The existence of $\sigma_D$ given $\hat{\sigma}$ is guaranteed by Theorem 3. Essentially, $\sigma_D$ provides the sequence of inputs $u^\ast$ as required by Problem 1. To illustrate the relationship between $\sigma_D$ and $\hat{\sigma}$, let $u_0^\ast, u_1^\ast, \ldots, u_{\ast-1}^\ast$ be the inputs which have been computed, and let $s_0^\ast, s_1^\ast, \ldots, s_{\ast}^\ast$ be the sequence of state generated by the inputs. The $t$-th control input $u_t^\ast$ is obtained by finding the minimum cost transition $(s_t^\ast, u_t^\ast, s_{t+1}^\ast)$, where $u_t^\ast \in U$ and $s_{t+1}^\ast \in S'$. The set $U$ is defined as $U = \hat{\sigma}([s_0^\ast]_{\equiv_X} [u_0^\ast]_{\equiv_U} \ldots [s_{\ast}^\ast]_{\equiv_X})$, and $S'$ is the union of all $S''$ such that $([s_t^\ast]_{\equiv_X}, U, S'')$ is a transition of $\hat{T}$. The inputs $u_t^\ast$ can be computed by solving a linear program when the cost function is linear and the equivalence classes are polyhedral sets.

In the beginning, when the partitioning is coarse, a winning strategy $\hat{\sigma}$ might not exist even if the underlying system $D$ has an optimal solution. However, if one continues to refine the grid, a winning strategy will exist if $D$ has an optimal solution, and its cost of winning will converge to the optimal cost. See Section VI-C for the proof. In addition, the algorithm can be terminated at a specific iteration based on applications and computational resources.

Algorithm 1 can in fact be instantiated to any class of hybrid systems. However, the computational complexity of the optimization problems that will need to be solved in the construction of the abstract system and the extraction of a winning strategy will depend on the class of dynamics and the type of the cost function. For a piecewise linear system with linear cost function, the maximization during the abstraction procedure is a linear program, because the partitions of $\equiv_X$ and $\equiv_U$ are polyhedral sets (grid elements). If computation resources are limited, the best suboptimal controller found with respect to the cost upper bound $J$ is guaranteed to generate a trajectory that satisfies the properties $\Pi$ and has cost no greater than $J$.

C. Analysis of Algorithm 1

This section analyzes the output of Algorithm 1, and shows that the suboptimal cost converges to the optimal cost if a robust optimal strategy exists. Note that even without the existence of a robust optimal strategy, we can still guarantee that the costs due to refinement are non-increasing.
Theorem 6. If there exists a robust optimal control \( u^\ast \) there is one path in \( \text{Paths}^{\ast} \) \( \{ \text{costs} \} \) that for all \( \epsilon > 0 \) such that \( B_{\epsilon_x}(x_t) \subseteq P_i \) for all \( t \in [k+1] \), where \( \phi(x_0, u) = \{ x_t \}_{t \in [k+1]} \), and \( PS(\phi(x_0, u)) = \{ P_i \}_{t \in [k+1]} \).

Lemma 7. There exist bounds \( M_x > 0 \) and \( M_u > 0 \) and constance \( c_1, c_2 \geq 0 \) that depend on \( PS(\phi(x_0, u^\ast)) \), such that for all \( \epsilon_x \in [0, M_x] \) and \( \epsilon_u \in [0, M_u] \), if \( x_0 \in B_{\epsilon_x}(x_0^\ast) \) and \( u_t \in B_{\epsilon_u}(u_t^\ast) \) \( \forall t \in [k] \), where \( u = \{ u_t \}_{t \in [k]} \) and \( \phi(x_0, u) = \{ x_t \}_{t \in [k+1]} \), then for all \( t \in [k] \),

\[
\| x_{t+1} - x_{t+1}^\ast \|_\infty \leq c_1 \epsilon_x + c_2 \epsilon_u,
\]

\[
PS(\phi(x_0, u)) = PS(\phi(x_0^\ast, u^\ast)).
\]

Definition 7. An input sequence \( u \) is said to be robust with respect to an initial state \( x_0 \) if there exists \( \epsilon_x > 0 \) such that \( B_{\epsilon_x}(x_t) \subseteq P_i \) for all \( t \in [k+1] \), where \( \phi(x_0, u) = \{ x_t \}_{t \in [k+1]} \), and \( PS(\phi(x_0, u)) = \{ P_i \}_{t \in [k+1]} \).

Let us denote the elements in the iteration of Algorithm 1 corresponding to a particular \( \epsilon \) as \( \bar{T}_\epsilon \) for \( \bar{T} \), \( \hat{x}_0^\epsilon \) for \( \hat{x}_0 \), \( J_\epsilon \), for \( J \), \( \hat{\sigma}_\epsilon \) for \( \hat{\sigma} \) and \( \sigma_\epsilon \) for \( \sigma_D \).

Theorem 6. If there exists a robust optimal control \( u^\ast \) with respect to \( x_0^\ast \) for Problem 1, the sequence of sub-optimal costs \( \{ J_{\epsilon_\sigma/2^i} \}_{t \in \mathbb{Z}^+} \) output by Algorithm 1 converges to the optimal cost \( J_{opt} = W(\bar{T}_D, x_0, \Pi) \). Furthermore, for each sub-optimal cost \( J_{\epsilon_\sigma/2^i} \), there exists a suboptimal winning strategy \( \sigma_{\epsilon_\sigma/2^i} \) with cost of winning \( J_{\epsilon_\sigma/2^i} \).

The rest of this section proves Theorem 6. Proofs of some lemmas are provided in the appendix to improve readability of this section. Henceforth, let \( u^\ast = \{ u_t^\ast \}_{t \in [k]} \) be a robust optimal control input sequence with respect ot \( x_0^\ast \) and \( \zeta^\ast = \phi(x_0^\ast, u^\ast) = \{ x_t^\ast \}_{t \in [k+1]} \) be the corresponding optimal trajectory for Problem 1. The proof also requires a special kind of strategy that ensures that there is a unique path conforming to this strategy, by choosing inputs that result in exactly one successor state (see Figure 2).

Definition 8. A chain strategy for a transition system \( T \) and an initial state \( s_0 \) is a strategy \( \sigma \in \text{Str}(T) \) such that there is one path in \( \text{Paths}_\sigma^{\ast}(T, s_0) \).

To prove Theorem 6, first, we show that for any trajectory whose initial state and inputs have a bounded deviation from that of the optimal trajectory, the trajectory itself will have a bounded deviation from the optimal trajectory.

Lemma 7. There exist bounds \( M_x > 0 \) and \( M_u > 0 \) and constance \( c_1, c_2 \geq 0 \) that depend on \( PS(\phi(x_0^\ast, u^\ast)) \), such that for all \( \epsilon_x \in [0, M_x] \) and \( \epsilon_u \in [0, M_u] \), if \( x_0 \in B_{\epsilon_x}(x_0^\ast) \) and \( u_t \in B_{\epsilon_u}(u_t^\ast) \) \( \forall t \in [k] \), where \( u = \{ u_t \}_{t \in [k]} \) and \( \phi(x_0, u) = \{ x_t \}_{t \in [k+1]} \), then for all \( t \in [k] \),

\[
\| x_{t+1} - x_{t+1}^\ast \|_\infty \leq c_1 \epsilon_x + c_2 \epsilon_u,
\]

\[
PS(\phi(x_0, u)) = PS(\phi(x_0^\ast, u^\ast)).
\]
This lemma implies that the error from the optimal state at any time is bounded linearly by the error from the initial state and the largest error of control inputs from the optimal ones. As $\epsilon_x$ and $\epsilon_u$ decrease to zero, the state error decreases to zero. Although the constants $c_1$ and $c_2$ depend on $t$, $t$ would not make the constants unbounded because $t$ is finite. Next, we show that the suboptimal cost of this trajectory is bounded.

**Lemma 8.** Given the cost function in Problem 2, there exist bounds $M_x > 0$ and $M_u > 0$ and constants $c_3, c_4 \geq 0$ such that for all $\epsilon_x \in [0, M_x]$ and $\epsilon_u \in [0, M_u]$, if $x_0 \in B_{c_x}(x_0^*)$ and $u_t \in B_{c_u}(u_t^*) \ \forall t \in [k]$, where $u = \{u_t\}_{t \in [k]}$ and $\zeta = \phi(x_0, u)$,

$$|W(\zeta) - W(\zeta^*)| \leq c_3 \epsilon_x + c_4 \epsilon_u,$$

$$PS(\phi(x_0, u)) = PS(\phi(x_0^*, u^*)).$$

This lemma states that given a continuous cost function, there will be a small neighborhood of the optimal trajectory in which the trajectories will go through the same partition sequence and difference in the cost is bounded and decreases to zero if $\epsilon_x$ and $\epsilon_u$ decrease to zero.

At this point, we have shown that the suboptimal cost is bounded by terms that depend on the input error and initial state error. Next, we show that given a specific cost sub-optimality, there exists a strategy that satisfies this cost error. In other words, we can construct an abstraction to give a chain strategy that satisfies a certain cost error bound.

**Lemma 9.** Given any $\delta > 0$, there exists a chain winning strategy $\sigma$ for some $\hat{T} = \text{Abs}(\mathcal{T}_D, \equiv_X, \equiv_U)$ such that

$$|W(\hat{T}, \sigma, x_0^*) - W(\mathcal{T}_D, x_0^*, \Pi)| \leq \delta$$

where $x_0^* = [x_0^*]_{\equiv X}$.

**Proof:** The broad idea will be to identify neighborhoods $N^x_t$ around $x^*_t$ and $N^u_t$ around $u^*_t$ such that $N^x_t$ is contained in the region of the partition containing $x^*_t$, all transitions from $N^x_t$ and $N^u_t$ lead to $N^x_{t+1}$ and the neighborhoods $N^x_t$ and $N^u_t$ are contained in $B_{M_x}(x^*_t)$ and $B_{M_u}(x^*_t)$. Further, we will ensure that the maximum cost of any transition from $N^x_t$ to $N^x_{t+1}$ using an input from $N^u_t$ is bounded. Then, by choosing $N^x_t$ and $N^u_t$ to be regions of $\equiv_X$ and $\equiv_Y$, we obtain a chain strategy in $\hat{T} = \text{Abs}(\mathcal{T}_D, \equiv_X, \equiv_U)$, where the only region of $\hat{T}$ reachable from the abstract state $N^x_t$ on input $N^u_t$ is $N^x_{t+1}$. Refer to Figure 2 for an illustration of the chain strategy.

Let $PS(\{x^*_t\}_t) = \{P_i\}_t$. We construct the sequence inductively, starting from $t = k + 1$ and moving backwards. Let $N^x_{t+1}$ be a grid cell of size $\epsilon_0/2^j$ that contains an open ball around $x^*_{t+1}$ which is contained in $P_i$. We can find such $N^x_{k+1}$ because of the robustness of the optimal control as defined in Definition 7. Assume we have computed $N^x_{t+1}, N^u_{t+1}, \ldots, N^x_{k+1}$. We show how to compute $N^x_t$ and $N^u_t$. Note that as long as $N^x_t$ and $N^u_t$ are contained in $B_{M_x}(x^*_t)$ and $B_{M_u}(x^*_t)$, all the transitions from $N^x_t$ on $N^u_t$ will end in $N^x_{t+1}$. Hence, let $N^x_t \subseteq B_{M_x}(x^*_t)$ and $N^u_t \subseteq B_{M_u}(x^*_t)$. By induction, under this construction, all executions from $N^x_0$ will be in $N^x_t$ after $t$ steps. This chain of neighborhoods gives us a chain strategy.

Further, when $N^x_t \subseteq B_{c_x}(x_0^*)$ and $N^u_t \subseteq B_{c_u}(x_0^*)$ where $\epsilon_x \in [0, M_x]$ and $\epsilon_u \in [0, M_u]$, the cost of the strategy is within $\delta$ of the optimal cost where $c_3 \epsilon_x + c_4 \epsilon_u \leq \delta$ for some constants $c_3$ and $c_4$ as given by Lemma 8. Thus,
\[ |W(\zeta) - W(\zeta^*)| \leq \delta \text{ for any path } \zeta \text{ starting in an } \epsilon_x \text{ ball around } x_0^* \]  
In addition, choose the \( N_j^x \) and \( N_j^u \) such that they correspond to an element of an \( \epsilon_0/2^i \) grid for some \( i \) (not necessarily the same \( i \) for all neighborhoods). Finally, define \( \equiv_X \) and \( \equiv_U \) such that the \( N_j^x \) and \( N_j^u \) are all equivalence classes of \( X \) and \( U \), respectively. Note that we need to ensure that for any \( i, j, N_j^x \) is the same as \( N_i^x \) or the two are disjoint, and, a similar condition for \( N_j^u \) holds. This condition can be easily ensured during the construction by picking small enough \( \epsilon \).

Lemma 9 guarantees a chain strategy. However, the partitions corresponding to the neighborhoods of \( N_j^x \) and \( N_j^u \) may not correspond to a uniform grid for any \( \epsilon \). Enumeration in Algorithm 1 only contains uniform grids with grid size \( \epsilon_0/2^i \). Thus, the next lemma constructs a uniform grid by refining the chain strategy obtained from Lemma 9.

**Lemma 10.** For a given \( \delta > 0 \), there exists an \( \epsilon = \epsilon_0/2^i > 0 \), such that \( |W(\mathcal{T}_e, x_0^*, \Pi) - W(\mathcal{T}_D, x_0^*, \Pi)| \leq \delta \), where \( x_0^* = [x_0^*]_{\equiv_X}. \) Furthermore, there exists a winning strategy \( \sigma_\epsilon \) with cost of winning \( W(\mathcal{T}_e, x_0^*, \Pi) \).

**Proof:** From the proof of Lemma 9, we obtain a sequence of neighborhoods \( N_i^x \) and \( N_i^u \) which correspond to a chain strategy, say \( \sigma \) starting from \( N_0^x \). Further, as observed in the proof, we can assume that every \( N_i^x \) corresponds to an element of \( \text{Grid}(X, \epsilon_0/2^i) \) for some \( i \), and similarly, \( N_i^u \) corresponds to an element of \( \text{Grid}(U, \epsilon_0/2^i) \) for some \( j \). Let \( i \) be the maximum of the \( i \)s and \( j \)s. Note that \( \text{Grid}(X, \epsilon_0/2^i) \) refines \( N_i^x \) and similarly, \( \text{Grid}(U, \epsilon_0/2^i) \) refines \( N_i^u \). In Figure 2, the squares around \( x_i^* \) with bold borders are \( N_i^x \), and the dashed squares which are contained in them correspond to the refined partition. One can define a strategy \( \sigma_\epsilon \) (not necessarily a chain anymore) for \( \mathcal{T}_e \) which correspond to following the neighborhoods \( N_i^x \). Hence, all the paths in \( \mathcal{T}_e \) which conform to \( \sigma_\epsilon \) will be contained in the neighborhoods \( N_i^x \). Therefore, the cost of \( \sigma_\epsilon \) is bounded by that of \( \sigma \) which is at most \( \delta \) away from the optimal cost. Therefore, the optimal cost of \( \mathcal{T}_e \) is at most \( \delta \) away from that of \( \mathcal{T}_D \).

**Proof of Theorem 6.** First, observe that \( J_{\epsilon_0/2^i} \leq J_{\epsilon_0/2^j} \) for all \( i > j \). Further, from Lemma 10, for any \( \delta > 0 \), there exists \( \epsilon = \epsilon_0/2^i \), such that \( |W(\mathcal{T}_e, x_0^*, \Pi) - W(\mathcal{T}_D, x_0^*, \Pi)| \leq \delta \). Note \( J_e = W(\mathcal{T}_e, x_0^*, \Pi) \) and \( J_{opt} = W(\mathcal{T}_D, x_0^*, \Pi) \) is the optimal cost. Hence, \( |J_e - J_{opt}| \leq \delta \). Therefore, \( J_{\epsilon_0/2^i} \) converges to \( J_{opt} \) as \( i \) goes to infinity. In addition, from Lemma 10, for each sub-optimal cost \( J_{\epsilon_0/2^i} \), there exists a suboptimal winning strategy \( \sigma_{\epsilon_0/2^i} \) with cost of winning \( J_{\epsilon_0/2^i} \).

At this point, we have shown that the strategy given by OptCAR incurs a suboptimal cost that converges to the optimal cost of \( \mathcal{D} \). The strategies used in the proof of Theorem 6 have the property that the length of the maximal paths which conform to the strategy are finite and have a bound (in fact, they are all of the same length). Further, the trace of all the paths is the same. However, during implementation, Algorithm 1 may return a sequence of suboptimal strategies \( \sigma_{\epsilon_0/2^i} \) that results in paths with different lengths. Nonetheless, the cost of each path results from \( \sigma_{\epsilon_0/2^i} \) is bounded by the cost \( J_{\epsilon_0/2^i} \).

In addition, the strategy that is considered in the proof of Theorem 6 gives a sequence of inputs which satisfy the property \( \Pi \) from any point in an open neighborhood around the given initial state \( x_0^* \). Further, there is an open neighborhood around each of the control inputs such that the resulting paths satisfy \( \Pi \). Hence, Algorithm 1 in fact returns a controller that is robust against input uncertainties under the assumption that the original system has such optimal control.
VII. OPTIMAL CONTROL OF FINITE TRANSITION SYSTEMS

This section presents a value iteration scheme for computing the optimal cost and optimal strategy for finite
transition systems. Observe that the strategies of the abstract system that are used in the proof of Theorem 6 have
a linear structure, that is, there are no paths in the abstract system of length greater than the number of states in the
system that conform with the strategy. We call such a strategy a layered strategy. Hence, in this section we present
an algorithm for computing an optimal strategy for a finite state transition system that is layered. The algorithm is
given in Algorithm 2 which is a modified Bellman-Ford algorithm [27].

The function ReduceReach reduces the problem of computing the layered strategy for a property \( \Pi \) to that of
reachability. It consists of taking a product of the input transition system \( T_S \) and the transition system \( T_P \) of the
property. More formally, given the input transition system \( T_S \) and the transition system \( T_P \) of the property,
the product transition system returned by ReduceReach is defined as follows.

**Definition 9.** Let \( T_S = (S_S, S^{\text{init}}_S, U_S, P, \Delta_S, L_S, W_S) \) be a state transition system, and \( T_P = (S_P, S^{\text{init}}_P, U_P, P, \Delta_P, L_P, W_P) \) be the automation that represents the regular property. Then, the product transition system is \( T = (S, S^{\text{init}}, U, P, \Delta, L, W) \) where

- \( S = \{ (s_1, s_2) \in S_S \times S_P \mid L_S(s_1) = L_P(s_2) \} \cup \{ s_d \} \) where \( s_d \) is a dead state;
- \( S^{\text{init}} = S^{\text{init}}_S \times S^{\text{init}}_P \);
- \( U = U_S \);
- \( P \) is the same for both \( T_S \) and \( T_P \). The final states of \( T_P \) is denoted by a proposition \( P_f \in P \);
- \( \Delta = \Delta_1 \cup \Delta_2 \), where \( \Delta_1 = \{ ((s_1, s_2), u, (s'_1, s'_2)) \in S \times U \times (S \setminus \{s_d\}) \mid (s_1, u, (s'_1) \in \Delta_S, (s_2, a, (s'_2)) \in \Delta_P \}

**Algorithm 2 SolveFiniteGame** (Two-Player Games)

**Require:** Finite state transition system \( T_S \), Property \( \Pi \) specified as \( (T_P, P_f) \)

\( T, S_f := \text{ReduceReach}(T_S, T_P, P_f) \)

Set for every \( s \in S - S_f \), \( C(s) := 0 \) if \( s \in S_f \) and \( \infty \) otherwise

for \( i = 1, \ldots, |S| \) do

for \( s \in S \) do

\[
C^i(s) := \min_{u \in U} \max_{(s,u,s') \in \Delta} (W(s,u,s') + C^{i-1}(s'))
\]

\[
\sigma^i(s) := \arg\min_{u \in U} \max_{(s,u,s') \in \Delta} (W(s,u,s') + C^{i-1}(s'))
\]

end for

end for

if \( C^{|S|}(s_0) < \infty \) then

Output the strategy \( \sigma^{|S|} \) and the cost \( C^{|S|}(s_0) \)

end if
for some \( a \) and \( \Delta_2 = \{(s_1, s_2, u, s_d) \in \mathcal{S} \times U \times \{s_d\} \} \) such that there exists \( (s_1, u, s_1') \in \Delta_S \) for some \( s_1' \) and there does not exist \( a \) and \( s_2' \) such that \( (s_2, a, s_2') \in \Delta_P \) and \( \mathcal{L}_S(s_1') = \mathcal{L}_P(s_2') \);

- \( \mathcal{L}(s) = \mathcal{L}_S(s) \) for \( s \in \mathcal{S}_S \);
- \( \mathcal{W}((s_1, s_2), u, (s_1', s_2')) = \mathcal{W}_S(s_1, u, s_2) \).

Furthermore, the set of final states \( S_f \) of \( T \) with respect with reachability is solved as \( S_f = \{(s_1, s_2) \in (\mathcal{S} \setminus \{s_d\}) \times (\mathcal{S} \setminus \{s_d\}) \mid \mathcal{L}_P(s_2) = P_f\} \).

The algorithm initially assigns a cost of 0 to the states in \( S_f \) and \( \infty \) otherwise. The cost \( C^i \) in the \( i \)-iteration captures the optimal cost of reaching \( S_f \) by a strategy in which all paths that conform to it have length at most \( i \), and \( \sigma^i \) stores a corresponding strategy. Hence, \( C^{\mathcal{L}}(s_0) \) provides a layered strategy if \( C^{\mathcal{L}}(s_0) < \infty \). The algorithm can be improved wherein it terminates earlier than completing the \( |\mathcal{S}| \) iterations, if the costs \( C \) do not change between iterations. In the worst case, this algorithm runs in \( O(|\Delta||\mathcal{S}|) \) time where \( |\Delta| \) is the number of transitions in \( T \) and \( |\mathcal{S}| \) is the number of states in \( \mathcal{T} \).

VIII. Implementation

Algorithm 1 and 2 are implemented in the tool OptCAR in Python 2.7. A Python package, NetworkX, is used to represent the graph structures that arise in solving Algorithm 2, and the Parma Polyhedra Library [28] is used to represent the polyhedral sets that arise in the gridding and to solve the linear program problem that arises in the weight computation. OptCAR is tested on a linear dynamical system and a piecewise linear system on a MacBook Pro 8.2, 4 core Intel Core i7 processor with speed 2200 Hz, and 8GB RAM.

A. Linear Dynamical System

The following linear dynamical system example is obtained from [29]:

\[
x_{t+1} = Ax_t + Bu_t
\]

where \( x_t = (x^1_t, x^2_t) \in [-1, 1]^2 \), and \( u_t \in [-1, 1] \).

The cost function is \( J(\phi(x_0, u)) = \sum_{t \in [k]} \|u_t\|_2^2 \). This cost is approximated as \( \sum_{t \in [k]} \|u_t\|_1 \) during implementation of OptCAR. The goal is to drive the system from an initial point \( x_0 = (0.9, 0.9) \) to a final zone defined by a box at the origin, \( P_f = \{ x \mid \|x\|_\infty \leq \frac{1}{2} \} \). The propositions of this example are represented in Figure 3.

![Fig. 3. Automaton that represents the propositions of the two examples where \( P_1 \) is the pink region, \( P_2 \) is the white region, and \( P_f \) is the light blue region.](image-url)
algorithm is implemented on two uniform grids on the states - $20 \times 20$ and $40 \times 40$. The input, $u$, is partitioned into 5 uniform intervals.

Strategies obtained from OptCAR are compared with the strategy given by linear quadratic regulator (LQR) in Table I. For a linear system, LQR is always a superior technique in comparison to OptCAR because the computation is significantly more efficient. The goal of this example is to illustrate that in an example with known optimal controller, the strategies given by OptCAR approximates the optimal control of LQR very closely, and it improves with refinement. Figure 4 shows the state trajectory for the three cases.

**Table I**

| Grid  | $20 \times 20$ | $40 \times 40$ | LQR |
|-------|----------------|----------------|-----|
| Computation time (seconds) | 355.82 | 5212.91 | 0.04 |
| Optimal cost | 0.5 | 0 | 0 |
| Optimal step | 6 | 6 | 6 |
| Final point | (-0.0468,0.1499) | (-0.0468,0.1999) | (-0.0468,0.1999) |

1 \(20 \times 20\) and \(40 \times 40\) are optimal controller synthesis using OptCAR for two different uniform grids. Computation time is the time a method takes to compute the optimal strategy. Optimal step is the total number of steps that the optimal path takes to reach the goal region. Final point is where the optimal path ends in the goal region.

Fig. 4. Simulated result of OptCAR and LQR on the linear dynamical system.
B. Two-tank System

A two-tank system from [30] is used as an example of a piecewise linear system (Figure 5). The water can flow in between the two tanks through a pipe that connects them. The pipe is located at level 0.2. Tank 1 (left) has an inflow of water that is managed by a controller, and tank 2 (right) has an outflow of water that is fixed. The goal of the controller is to fill up tank 2 to level 0.4 from an initially low water level 0.1 using as small amount of water as possible from the source above tank 1. The goal will be made precise after the system is described formally next.

The two-tank system has a linearized dynamics given by

\[ x_{t+1} = Ax_t + Bu_t \]  \hfill (2)

where

\[ A = \begin{cases} 
A_1 & x \in [0, 0.2]^2 \\
A_2 & \text{otherwise}
\end{cases}, 
B = \begin{bmatrix} 342.6753 \\
0 \end{bmatrix}, \]

\[ A_1 = \begin{bmatrix} 1 & 0 \\
0 & 0.9635 \end{bmatrix}, 
A_2 = \begin{bmatrix} 0.8281 & 0.1719 \\
0.1719 & 0.7196 \end{bmatrix}, \]

\[ x_t = (x_1^t, x_2^t) \in [0, 0.7]^2, \text{ and } u_t \in [0, 0.0005]. \] The water level in tank 1 at time \( t \) is \( x_1^t \), and the water level in tank 2 at time \( t \) is \( x_2^t \). The cost function is chosen to be \( J(\phi(x_0, u)) = \sum_{t \in [k]} |u_t|_1 \) to represent minimal water inflow, and the goal is to drive the system from partition \([0, 0.7] \times [0, 0.1]\), to partition \([0, 0.7] \times [0.4, 0.7]\). The propositions of this example are represented in Figure 3.

The algorithm is implemented on two uniform grids on the states - \( 28 \times 17 \) (coarse) and \( 56 \times 33 \) (refined), and two non-uniform grids - \( 23 \times 17 \) (coarse) and \( 32 \times 25 \) (refined). Figure 6 illustrates the grids. The input, \( u \), is partitioned into 10 uniform intervals. The goal region is represented as one partition for all cases. This choice of goal representation speeds up computation time, and does not change the results in Section VI-C. Once a path arrives at the goal region, the path ends. Thus, the goal region does not need to be partitioned because the transitions within the goal region is irrelevant. In addition, the partitions’ sizes for a non-uniform grid do not necessary have

![Fig. 5. A schematic of a two-tank system.](image-url)
(a) Uniform $28 \times 17$

(b) Uniform $56 \times 33$

(c) Non-uniform $23 \times 17$

(d) Non-uniform $38 \times 25$

Fig. 6. Partitions for the two-tank system. The goal region does not need to be partitioned because the transitions within the goal region are irrelevant.

to be the same. Partitions whereby the transitions are more likely to be far can be larger because the states most likely will not end up at the neighboring partitions if the partitions are small. Another example of non-uniform grids with the same principle would be to have finer grids near the goal and coarser grids away from the goal. Such modification is feasible if the control engineer has prior information about the system from his/her past experiences. These modifications reduce computation time, and also allow for finer grids at regions that matter to achieve a better result.

Strategies obtained from OptCAR are compared in Table II. Figure 7 shows the state trajectory and control input for the four cases, all start from $(0.001, 0.001)$. This example shows that choosing a suitable partition can reduce the computation time dramatically while still achieving comparable performance to the performance of a naive uniform grid. Hence, future extension of this technique includes designing an intelligent scheme to partition

(a) State trajectory

Fig. 7. State trajectory and control input generated by the controller from OptCAR for the two-tank system.
TABLE II
PERFORMANCE OF OPTCAR USING DIFFERENT GRIDS FOR SYSTEM (2). ²

| Grid       | $28 \times 17$ | $56 \times 33$ | $23 \times 17$ | $38 \times 25$ |
|------------|----------------|----------------|----------------|----------------|
| Computation time (seconds) | 1234.53 | 22119.36 | 1057.43 | 4309.04 |
| Optimal cost | 0.00340 | 0.00320 | 0.00335 | 0.00320 |
| Optimal step | 12 | 12 | 12 | 13 |
| Final point | $(0.642,0.402)$ | $(0.573,0.401)$ | $(0.625,0.405)$ | $(0.552,0.412)$ |

² The first two columns are results for uniform grids. The last two columns are results for non-uniform grids. The grids are shown in Figure 6. Computation time is the time OPTCAR takes to compute the optimal strategy. Optimal step is the total number of steps that the optimal path takes to reach the goal region. Final point is where the optimal path ends in the goal region.

the domain such that computation time is reduced. Lastly, about 60% – 70% of the computation time are used to construct the abstraction (i.e. ConsAbs step in Algorithm 1) in which the computations can be parallelized easily to decrease computation time.

IX. CONCLUSION

In this paper, we consider the problem of synthesizing optimal control strategies for discrete-time piecewise linear system with respect to regular properties. We present an abstraction-refinement approach for constructing arbitrarily precise approximations of the optimal cost and the corresponding strategies. This approach computes a sequence of suboptimal controller that converges to the optimal controller with refinement. The resulting suboptimal controller would generate trajectories that incurs cost no greater than the optimal cost of the corresponding abstract system. The abstraction based approach can be applied to the general class of hybrid systems and for properties over infinite traces, however, the challenge is in computing edges and weights, especially, for non-linear dynamics and in continuous time.

Future work will include extending the technique to more complex dynamics and continuous-time hybrid systems. In addition, the cost preserving abstraction technique will be extended from regular properties to $\omega$-regular properties. To reduce computation time, a more intelligent gridding scheme in the refinement step will be developed. Lastly, the neighborhoods of states and inputs in the abstract system naturally model measurement errors and input uncertainties of the concrete system. Hence, a potential future application of this technique is in synthesizing robust optimal control for a hybrid system.

ACKNOWLEDGMENT

Pavithra Prabhakar was partially supported by EU FP7 Marie Curie Career Integration Grant No. 631622 and NSF CAREER 1552668. This work was partially conducted when Yoke Peng Leong was an intern at IMDEA Software Institute.
APPENDIX A

PROOF FOR LEMMA 7

Rewrite $x_{t+1} = A_t x_t + B_t u_t$ as

$$x_{t+1} = \left( \prod_{j=0}^{t} A_j \right) x_0 + B_t u_t + \sum_{k=1}^{t-1} \left( \prod_{j=k}^{t} A_j \right) B_{k-1} u_{k-1}$$

where $\prod_{j=0}^{t} A_j = A_t A_{t-1} \ldots A_1 A_0$. Then,

$$\|x_{t+1} - x_{t+1}^*\|_\infty = \left\| \left( \prod_{j=k}^{t} A_j \right) (x_0 - x_0^*) + B_t (u_t - u_t^*) + \sum_{k=0}^{t-1} \left( \prod_{j=k}^{t} A_j \right) B_{k-1} (u_{k-1} - u_{k-1}^*) \right\|_\infty$$

$$\leq \prod_{j=0}^{t} \|A_j\|_\infty \|x_0 - x_0^*\|_\infty + \|B_t\|_\infty \|u_t - u_t^*\|_\infty + \sum_{k=1}^{t} \left( \prod_{j=k}^{t} \|A_j\|_\infty \right) \|B_{k-1}\|_\infty \|u_{k-1} - u_{k-1}^*\|_\infty$$

$$= c_1 \epsilon_x + c_2 \epsilon_u$$

where $c_1 = \prod_{j=0}^{t} \|A_j\|_\infty$ and $c_2 = \|B_t\|_\infty + \sum_{k=1}^{t} \left( \prod_{j=k}^{t} \|A_j\|_\infty \right) \|B_{k-1}\|_\infty$.

APPENDIX B

PROOF FOR LEMMA 8

First, compute

$$|\mathcal{W}(\xi) - \mathcal{W}(\xi^*)| = \left| \sum_{t=0}^{k} J(x_{t+1}, u_t) - J(x_{t+1}^*, u_t^*) \right|$$

$$\leq \sum_{t=0}^{k} |J(x_{t+1}, u_t) - J(x_{t+1}^*, u_t^*)|.$$ 

Since $J(x, u)$ is a continuous function, there exists a constant $C_t > 0$ such that

$$|J(x_{t+1}, u_t) - J(x_{t+1}^*, u_t^*)| \leq C_t \|\bar{x}_t - \bar{x}_t^*\|_\infty.$$ 

Hence,

$$|\mathcal{W}(\xi) - \mathcal{W}(\xi^*)| \leq \sum_{t=0}^{k} C \|\bar{x}_t - \bar{x}_t^*\|_\infty$$

where $C = \max_{t \in [k]} C_t$ and $\bar{x}_t = [x_{t+1}, u_t] \in \mathbb{R}^{n+p}$ is a joined vector of $x$ and $u$. By Lemma 7,

$$|\mathcal{W}(\xi) - \mathcal{W}(\xi^*)| \leq \sum_{t=0}^{k} C \max\{c_1 (t+1) \epsilon_x + c_2 (t+1) \epsilon_u, \epsilon_u\}$$

$$\leq \max\{c_3' \epsilon_x + c_4' \epsilon_u, kC \epsilon_u\}$$
where \( c_3' = \max_{t \in [k]} kCc_1(t + 1) \), and \( c_4' = \max_{t \in [k]} kCc_2(t + 1) \). Then,

\[
|W(\zeta) - W(\zeta^*)| \leq c_3\epsilon_x + c_4\epsilon_u
\]

\[
c_3 = \begin{cases} 
    c_3', & \epsilon_x c_3' + \epsilon_u c_4' \geq k\epsilon_u \\
    0, & \epsilon_x c_3' + \epsilon_u c_4' < k\epsilon_u 
\end{cases}
\]

\[
c_4 = \begin{cases} 
    c_3', & \epsilon_x c_3' + \epsilon_u c_4' \geq k\epsilon_u \\
    kC, & \epsilon_x c_3' + \epsilon_u c_4' < k\epsilon_u 
\end{cases}
\]

REFERENCES

[1] A. Church, “Logic, arithmetic, and automata,” in International Congress of Mathematicians, 1962, pp. 23–35.

[2] Z. Manna and P. Wolper, “Synthesis of communicating processes from temporal logic specifications,” in Logics of Programs, Workshop, Yorktown Heights, New York, May 1981, pp. 253–281.

[3] E. M. Clarke and E. A. Emerson, “Design and synthesis of synchronization skeletons using branching-time temporal logic,” in Logics of Programs, Workshop, Yorktown Heights, New York, May 1981, pp. 52–71.

[4] P. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” SIAM J. Control Optim., vol. 25, no. 1, pp. 206–230, Jan 1987.

[5] E. Asarin, O. Maler, and A. Pnueli, “Symbolic controller synthesis for discrete and timed systems,” in Hybrid Systems II, 1994, pp. 1–20.

[6] T. A. Henzinger, B. Horowitz, and R. Majumdar, “Rectangular hybrid games,” in CONCUR ’99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 1999, pp. 320–335.

[7] P. Bouyer, T. Brihaye, and F. Chevalier, “O-minimal hybrid reachability games,” Logical Methods in Computer Science, vol. 6, no. 1, 2010.

[8] V. Vladimiroiu, P. Prabhakar, M. Viswanathan, and G. E. Dullerud, “Specifications for decidable hybrid games,” Theor. Comput. Sci., vol. 412, no. 48, pp. 6770–6785, 2011.

[9] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata?” in Twenty-seventh Annual ACM Symposium on Theory of Computing, ser. STOC ’95, 1995, pp. 373–382.

[10] J. Raisch and S. O’Young, “Discrete approximation and supervisory control of continuous systems,” IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems, vol. 43, no. 4, pp. 569–573, 1998.

[11] T. Wongpiromsarn, U. Topcu, and R. M. Murray, “Receding horizon temporal logic planning,” IEEE Transactions on Automatic Control, vol. 57, no. 11, pp. 2817–2830, 2012.

[12] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, “Synthesis of reactive switching protocols from temporal logic specifications,” IEEE Transactions on Automatic Control, vol. 58, no. 7, pp. 1771–1785, 2013.

[13] M. Kloetzer and C. Belta, “Temporal logic planning and control of robotic swarms by hierarchical abstractions,” IEEE Transactions on Robotics, vol. 23, no. 2, pp. 320–330, 2007.

[14] J. A. DeCastro and H. Kress-Gazit, “Guaranteeing reactive high-level behaviors for robots with complex dynamics,” in IEEE Int. Conf. on Intelligent Robots and Systems (IROS), Nov 2013, pp. 749–756.

[15] C. Seatzu, D. Gromov, J. Raisch, D. Corona, and A. Giua, “Optimal control of discrete-time hybrid automata under safety and liveness constraints,” Nonlinear Analysis: Theory, Methods & Applications, vol. 65, no. 6, pp. 1188 – 1210, 2006.

[16] E. A. Gol, M. Lazar, and C. Belta, “Temporal logic model predictive control,” Automatica, vol. 56, pp. 78 – 85, 2015.

[17] A. Bemporad, F. Borrelli, and M. Morari, “Piecewise linear optimal controllers for hybrid systems,” in American Controls Conf. (ACC), vol. 2, 2000, pp. 1190–1194.

[18] B. Lincoln and B. Bernhardsson, “LQR optimization of linear system switching,” IEEE Transactions on Automatic Control, vol. 47, no. 10, pp. 1701–1705, 2002.

[19] A. Girard, “Controller synthesis for safety and reachability via approximate bisimulation,” Automatica, vol. 48, no. 5, pp. 947–953, 2012.

[20] G. Reissig and M. Rungger, “Abstraction-based solution of optimal stopping problems under uncertainty,” in IEEE Int. Conf. on Decision and Control (CDC), 2013, pp. 3190–3196.

[21] M. Mazo and P. Tabuada, “Symbolic approximate time-optimal control,” Systems & Control Letters, vol. 60, no. 4, pp. 256–263, 2011.
[22] S. Karaman, R. G. Sanfelice, and E. Frazzoli, “Optimal control of mixed logical dynamical systems with linear temporal logic specifications,” in IEEE Int. Conf. on Decision and Control (CDC), 2008, pp. 2117–2122.

[23] E. M. Wolff, U. Topcu, and R. M. Murray, “Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic,” in IEEE Int. Conf. on Decision and Control (CDC), 2013, pp. 3197–3204.

[24] Y. P. Leong and P. Prabhakar, “Optimal control with regular objectives using an abstraction-refinement approach,” in American Controls Conf. (ACC), 2016, pp. 5161–5168.

[25] M. Y. Vardi and P. Wolper, “An automata-theoretic approach to automatic program verification (preliminary report),” in Symposium on Logic in Computer Science, Jun 1986, pp. 332–344.

[26] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi, “Alternating refinement relations,” in Ninth International Conference on Concurrency Theory (CONCUR’98), vol. 1466, 1998, pp. 163–178.

[27] R. Bellman, “On a routing problem,” DTIC Document, Tech. Rep., 1956.

[28] R. Bagnara, P. M. Hill, and E. Zaffanella, “The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems,” Science of Computer Programming, vol. 72, no. 1–2, pp. 3–21, 2008.

[29] Y. Tazaki and J. Imura, “Discrete abstractions of nonlinear systems based on error propagation analysis,” IEEE Transactions on Automatic Control, vol. 57, no. 3, pp. 550–564, March 2012.

[30] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Bela, “Temporal logic control of discrete-time piecewise affine systems,” IEEE Transactions on Automatic Control, vol. 57, no. 6, pp. 1491–1504, June 2012.

Yoke Peng Leong received the B.S. and M.S. degrees in Mechanical Engineering from Northwestern University, Evanston, IL, USA, in 2012. She is currently a Ph.D. candidate in Control and Dynamical Systems at the California Institute of Technology, Pasadena, CA, USA.

Pavithra Prabhakar is an associate professor of computer science at Kansas State University, where she holds the Peggy and Gary Edwards Chair in Engineering. She obtained her doctorate in Computer Science from the University of Illinois at Urbana-Champaign (UIUC) in 2011, from where she also obtained a masters in Applied Mathematics. She was a CMI (Center for Mathematics of Information) fellow at Caltech for the year 2011-12. She has been on the faculty of Kansas State University since 2015, and has previously held a faculty position at the IMDEA Software Institute. Her main research interest is in the Formal Analysis of Cyber-Physical Systems, with emphasis on both theoretical and practical methods for verification and synthesis of hybrid control systems. Her papers have been selected for a best paper honorable mention award from Hybrid Systems: Computation and Control, best papers of MEMOCODE and invited papers at Allerton and American Control Conference. She has been awarded a Sohaib and Sara Abbasi fellowship from UIUC, an M.N.S Swamy medal from the Indian Institute of Science for the best masters thesis, a Marie Curie Career Integration Grant from the European Union, Michelle Munson-Serban Sinu Keystone Research Faculty Scholarship from the KSU College of Engineering, a summer faculty fellowship from AFRL, an NSF CAREER Award and an ONR Young Investigator Award.