Temporal Logic Control of Switched Affine Systems with an Application in Fuel Balancing

Petter Nilsson, Necmiye Özay, Ufuk Topcu and Richard M. Murray

***Abstract*— We consider the problem of synthesizing hierarchical controllers for discrete-time switched affine systems subject to exogenous disturbances that guarantee that the trajectories of the system satisfy a high-level specification expressed as a linear temporal logic formula. Our method builds upon recent results on temporal logic planning and embedded controller synthesis. First, the control problem is lifted to a discrete level by constructing a finite transition system that abstracts the behavior of the underlying switched system. At the discrete level, we recast the problem as a two player temporal logic game by treating the environment driven switches as adversaries. The solution strategy for the game (i.e. the discrete plan) is then implemented at the continuous level by solving finite-horizon optimal control problems that establish reachability between discrete states and that compensate the effects of continuous disturbances. We also extend the earlier work by making efficient use of propositions in the temporal logic formula to drive the abstraction procedure and to facilitate the computation of continuous input at implementation time.**

An aircraft fuel system example is formulated; and solved using the proposed method. This sample problem demonstrates the applicability of the abstraction procedure and correct-by-construction controllers to regulate the fuel levels in multiple tanks during interesting operations like aerial refueling.

I. INTRODUCTION

Temporal logics provide a formal means to specify and verify the correct behavior of systems and have been extensively used in digital circuit design and software engineering [1], [2]. Due to their expressive power, there has been a recent interest in the use of temporal logics as specification languages for dynamical systems (see, for instance, [3], [4], [5], [6]). Leveraging recent results on temporal logic planning and embedded controller synthesis [7], [8], [9], in this paper we present a method for automatic synthesis of controllers for switched affine systems. The controllers are *correct-by-construction* in the sense that any execution of the system is guaranteed to fulfill a given linear temporal logic (LTL) specification.

In order to enable discrete planning we lift the problem to the discrete level, which requires computing a finite transition system. The ability to construct a controller depends on how many possible transitions are established in this finite transition system. The first contribution of the present paper is to enhance the performance of the discretization method proposed by Wongpiromsarn et al. [7] by relaxing the requirements for allowed transitions between discrete states, while still ensuring system correctness. By allowing more transitions it is possible to synthesize controllers for a wider range of systems and at the same time get a system performance that is closer to optimal.

Our second contribution is the extension of temporal logic planning framework to systems whose dynamical mode can be changed by the environment, so called *switched* systems. Control of switched systems is a challenging problem and has attracted considerable attention [10]. Although a thorough survey is beyond the scope of this paper, we give a very brief overview of existing work. The main body of results is on stability analysis for certain classes of switching [11], [12], [13]. Blanchini et al. [14] provide a characterization of all stabilizing controllers for linear switched systems under arbitrary switching. Tarraf et al. [15] use finite abstractions to design stabilizing switching controllers for second order systems. We also use finite abstractions, but instead of stability we consider controller design for an expressive subset of LTL specifications. Our framework can handle arbitrary switching. Moreover, it naturally allows to specify a priori assumptions on the switching sequence. For instance, it is possible to accommodate different time scales for the switch sequence and the system dynamics or to restrict certain transitions among different modes if such a priori information is available.

Finally, motivated by design challenges in vehicle management systems (VMS) that control and coordinate a number of subsystems of aerial vehicles (e.g., flight controllers, electrical systems, environmental control systems, fuel systems, deicing units, and landing gear [16], [17], [18]), we apply the proposed method to a VMS example. In particular, we demonstrate how a switched system can be controlled with the proposed technique by synthesizing a controller for the fuel subsystem. The system is switched between normal operation, where fuel is consumed at a constant rate, and aerial refueling mode, where the net inflow of fuel to the system is positive. The goal is to balance the fuel volumes in different tanks and to ensure safe operation under all possible switching sequences in a prespecified class.

The rest of the paper is organized as follows. In section II, we introduce some definitions and the formal problem setting. Section III details the proposed hierarchical approach. We present the fuel balancing case study in section IV. Finally, section V concludes the paper with some remarks and directions for future research.

This work was supported in part by Caltech Summer Undergraduate Research Fellowship, the FCRP consortium through the Multiscale Systems Center (MuSyC) and the Boeing Corporation.

P. Nilsson is with the Royal Institute of Technology (KTH), Sweden. email: pettni@kth.se

N. Ozay, U. Topcu and R. M. Murray are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA. emails: {necmiye, utopcu, murray}@cds.caltech.edu

II. PRELIMINARIES AND PROBLEM FORMULATION

This section contains some definitions and background information that will be used throughout the paper, as well as a formal definition of the problem considered.

A. Polytopes

A *polytope* P is a convex set in $\mathbb{R}^n$, usually defined as the intersection of a finite number of half planes as $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. The projection of a polytope onto its first m coordinates is the set $proj_m(P) = \{x \in \mathbb{R}^m \mid \exists y \in \mathbb{R}^{n-m} : \begin{bmatrix} x & y \end{bmatrix} \in P\}$. Several algorithms exist for computing the projection of polytopes, the optimal choice of algorithm depends on the characteristics of the polytope. In this paper polytopes are used to define cells in a continuous state space.

We say that two disjoint polytopes $P_1 = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and $P_2 = \{x \in \mathbb{R}^n \mid Cx \leq d\}$ are *order 1 neighbors* if the set $\{x \in \mathbb{R}^n \mid [A]_i x \leq [b]_i + \varepsilon\} \cap \{x \in \mathbb{R}^n \mid [C]_j x \leq [d]_j + \varepsilon\}$ is non-empty for some choice of inequalities $[A]_i x \leq [b]_i$ and $[C]_j x \leq [d]_j$ from the two polytopes. We generalize the concept of neighbors by calling neighbors of neighbors *order 2 neighbors*, and so on.

B. Environment Switched Systems

We consider a switched system that has N_d different dynamical modes it can switch among, and assume that each such dynamic mode $\mathcal{M}_k$ is described by an affine dynamical model of the following form:

$$\mathcal{M}_k : \begin{cases} s(t+1) = A_k s(t) + B_k u(t) + E_k d(t) + K_k, \\ u(t) \in U_k(s(t)), \\ d(t) \in D_k, \end{cases} \tag{1}$$

where $U_k(s(t))$ and D_k are polytopes containing possible values for input and disturbance, respectively. In particular, $u(t) \in U_k(s(t))$ constitutes inequality constraints linear in $u(t)$ and $s(t)$ that represents the possibility of having the allowable input values depend on the state. We denote the dimension of the state space by n and the dimension of the input space by m. The vector K_k, the affine offset term, has dimension n and describes constant changes in the state.

An *environment switched* system is a system where the dynamical mode is controlled by the environment (i.e., the switches are uncontrollable). Assume that the environment controls a variable e_d, and that the dynamics of the plant depend on the value of e_d. As an example, when $e_d = 1$ the motion could be described by the dynamic mode $\mathcal{M}_1$, when $e_d = 2$ it could be described by $\mathcal{M}_2$, and so on.

C. Linear Temporal Logic

Linear Temporal Logic (LTL) is an extension of the classical logic by including temporal operators. Apart from the usual operators negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\rightarrow$), it provides the possibility to write statements including the temporal operators next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$). This makes it possible to write a wide range of requirements on the desired behavior of a system.

Definition 1: An *atomic proposition* is a statement on a system variable v that has a unique truth value (*True* or *False*) for a given value of v.

Given a set Π of atomic propositions, an LTL formula is defined inductively as follows: (i) any atomic proposition $p \in \Pi$ is an LTL formula; and (ii) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas. Formulas involving other operators can be derived from these basic ones.

We consider LTL specification for a system $\mathbb{S}$ consisting of a plant and its environment. The set of system variables in $\mathbb{S}$ are $V = S \cup E$, where S is the set of variables controlled by the plant, i.e. the variables in $s(t)$ in (1), and E is the set of variables controlled by the environment. The set of possible states in $\mathbb{S}$ is therefore $dom(V) = dom(S) \times dom(E)$ and every state v of the system can be written $v = (s, e)$ where $s \in dom(S)$ and $e \in dom(E)$. Here, $dom(V)$ stands for the *domain* of V, i.e., the set of valuations of V.

An *execution* σ of $\mathbb{S}$ is an infinite sequence of its states, i.e., $\sigma = (s(0), e(0)), (s(1), e(1)), \ldots$. LTL formulas are interpreted over state sequences. We refer the reader to [1] for exact semantics of LTL. We say φ is satisfied by $\mathbb{S}$ if φ is true for all executions of $\mathbb{S}$.

D. Problem Formulation

We consider a system $\mathbb{S}$ together with some specification φ on the system, given in the form

$$\varphi \doteq (\varphi_e \rightarrow \varphi_s), \tag{2}$$

written in LTL. Generally speaking, some *assumptions* φ_e on the environment should ensure that the *requirements* φ_s on the plant states are satisfied. We denote by Π the set of atomic propositions from φ on the variables in V. In our setup, the propositions on the plant states $s(t)$ are in the form of linear inequality constraints that define convex polytopes corresponding to the regions of interest in the state space. Environment assumptions, for instance, can be used to define the a priori information on the possible switching sequences. Next, we give a formal definition of the problem.

Problem 1: Given a system $\mathbb{S}$ (consisting of an environment switched system (1) and a set of environment variables) and an LTL specification, φ, of the form (2), find a control input $u(t)$ such that $\mathbb{S}$ satisfies φ (i.e., whenever the assumptions on the environment hold, the trajectories of (1) will satisfy the requirements).

III. HIERARCHICAL APPROACH

The automatic planner/controller synthesis proposed in [6], [7], [8], [19] relies on several steps. In this paper, we adopt a similar hierarchical approach. For the sake of completeness, we present the overall methodology, highlighting the contributions required for reducing conservatism and for extending the results to switched systems.

Since the planner synthesis requires a finite number of states, it is necessary to construct a *finite transition system* $\mathbb{D}$. The finite transition system contains a partition of $dom(S)$ and $dom(E)$ into a finite number of equivalence classes (or

cells) $\mathcal{S}$ and $\mathcal{E}$. The resulting finite partition is denoted $\mathcal{V} = \mathcal{S} \times \mathcal{E}$. In the following we will call $s \in dom(S)$ a *continuous state* for the plant and $\varsigma \in \mathcal{S}$ a *discrete state*.

First, we partition the continuous state space $dom(S)$ into a coarse partition that is *proposition preserving*, and let each cell in the partition represent a discrete state. The partition is said to be proposition preserving if, for any discrete state in the partition, exactly the same atomic propositions in Π are true for all continuous states inside the discrete state. We also make sure that the cells in the partition are convex to be able to do convex optimization calculation of input signals, as explained in III-D. Any non-convex cell can easily be divided into several convex cells, and it is trivial to show that if the larger non-convex cell is proposition preserving, the smaller convex cells will also be so. We call this initial proposition preserving partition $\mathcal{P}$.

Second, we need the possible transitions in the finite transition system, for each dynamical mode $\mathcal{M}_k$. That is, we need N_d lists of possible transitions between the discrete states in the partition $\mathcal{P}$. The transitions are established through the concept of *reachability*, which is developed in III-A. In general it will not be possible to establish enough reachability relations between discrete states in the initial proposition preserving partition $\mathcal{P}$, since it is too coarse. Therefore further discretization has to be done to obtain a finer partition. We present in III-B a procedure that refines the partition in order to establish reachability relations. The refined partition together with the lists of possible transitions is the finite transition system $\mathbb{D}$.

Finally we synthesize a discrete planner for $\mathbb{D}$ and implement the planner by using a continuous controller that moves the plant between the discrete states, as discussed in III-C and III-D, respectively. The overall controller has a feedback structure that takes into account both the current state $s(t)$ and the current environment dependent mode k (which also depends on t).

A. Reachability

Assume that $\mathcal{S} = \{\varsigma_0, \varsigma_1, \dots \varsigma_n\}$ is a partition of the continuous state space $dom(S)$. We define the mapping $T_{\mathcal{S}} : dom(S) \to \mathcal{S}$ that takes a continuous state to its corresponding discrete state. The inverse mapping $T_{\mathcal{S}}^{-1}(\varsigma_i)$ describes the set of all continuous states that are inside the discrete state ς_i.

We also assume that there exists another proposition preserving partition $\mathcal{P} = \{\rho_0, \rho_1, \dots \rho_m\}$ of $dom(S)$, such that $\mathcal{S}$ is a refinement of $\mathcal{P}$. In other words, for every $\varsigma_i \in \mathcal{S}$ there exists a $\rho_j \in \mathcal{P}$ such that $T_{\mathcal{S}}^{-1}(\varsigma_i) \subseteq T_{\mathcal{P}}^{-1}(\rho_j)$, where $T_{\mathcal{P}} : dom(S) \to \mathcal{P}$ is defined in accordance with $T_{\mathcal{S}}$. The two partitions $\mathcal{S}$ and $\mathcal{P}$ may be equal.

Now we define the mapping $P_{\mathcal{S}} : \mathcal{S} \to \mathcal{P}$ that takes a discrete state to the cell in the initial proposition preserving partition from which it originates, i.e. $P_{\mathcal{S}}(\varsigma_i) = \rho_j \Leftrightarrow T_{\mathcal{S}}^{-1}(\varsigma_i) \subseteq T_{\mathcal{P}}^{-1}(\rho_j)$. Then we can define our idea of reachability in a fixed horizon length N, with respect to some dynamical mode $\mathcal{M}_k$.

Definition 2: A discrete state ς_j is *reachable* from another discrete state ς_i in N steps if, starting from *any* continuous state $s(0) \in T_{\mathcal{S}}^{-1}(\varsigma_i)$, there exists a control sequence $u(0), u(1), \dots u(N-1)$ that takes the plant to *some* continuous state $s(N) \in T_{\mathcal{S}}^{-1}(\varsigma_j)$ in accordance with the system dynamics $\mathcal{M}_k$. This should be possible for all disturbance sequences $d(0), d(1), \dots d(N-1) \in D_k^N$. Furthermore, we require that $u(t) \in U_k(s(t))$ for times $t \in \{0, 1, \dots N-1\}$, and that $s(t) \in T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$ for times $t \in \{1, 2, \dots N-1\}$.

The last requirement in the definition makes sure that the trajectory of the plant remains inside the proposition preserving cell, which is required to ensure correctness when implementing a correct discrete plan. Roughly speaking, the same set of atomic propositions will be true during the whole trajectory from ς_i to ς_j, which lets the continuous implementation inherit correctness from the discrete plan.

The partition refinement technique in the next section relies on solving the following problem:

1) The reachability problem: Given an initial discrete state ς_i and a final discrete state ς_j, find the set $S_0 \subseteq \varsigma_i$ such that ς_j is reachable from S_0.

2) Solving the reachability problem: By using the dynamics (1) and assuming that the dynamical mode stays constant up to time $t-1$, it can be shown that the plant state at time t can be written as $s(t) = A_k^t s(0) + \sum_{i=0}^{t-1} \left(A_k^i B_k u(t-1-i) + A_k^i E_k d(t-1-i) + A_k^i K_k \right)$ for matrices A_k, B_k, E_k and K_k corresponding to the dynamical mode $\mathcal{M}_k$. That is, the state at time t depends on the dynamical mode, the initial state and all disturbances and input signals up to time $t-1$. From the definition of reachability, we can write down the following constraints for a transition from ς_i to ς_j in N steps:

- $s(0) \in T_{\mathcal{S}}^{-1}(\varsigma_i)$,
- $s(t) \in T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$ for all $t \in \{1, 2, \dots N-1\}$,
- $s(N) \in T_{\mathcal{S}}^{-1}(\varsigma_j)$,
- $u(t) \in U_k(s(t))$ for all $t \in \{0, 1, \dots N-1\}$.

We assume that $T_{\mathcal{S}}^{-1}(\varsigma_i)$, $T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$, $T_{\mathcal{S}}^{-1}(\varsigma_j)$ and $U_k(s(t))$ are all polytopes, i.e. there exists L_0, L_t, L_N, L_U and M_0, M_t, M_N, M_U such that for example $T_{\mathcal{S}}^{-1}(\varsigma_i) = \{x \in \mathbb{R}^n \mid L_0 x \leq M_0\}$. By expressing the states as functions of previous inputs and disturbances, it turns out that all the constraints above can be stacked and expressed as a system of inequalities on the form $L \begin{bmatrix} s(0)^T & u(0)^T & u(1)^T & \dots & u(N-1)^T \end{bmatrix}^T \leq M - \hat{d}$, a set that defines a polytope of dimension $n + Nm$. This polytope contains all combinations of initial states and input sequences for which the reachability problem is feasible. The matrix L is built from A_k, B_k, L_0, L_t, L_N and L_U, the vector M from M_0, M_t, M_N and M_U and the matrix G from A_k, E_k, L_0, L_t, L_N and L_U. An element $[\hat{d}]_i$ in the vector $\hat{d}$ represents, for the corresponding inequality in the polytope, the effect of the *worst possible* disturbance sequence for that inequality. Considering the worst possible disturbances ensures that the reachability relation holds for all possible disturbances. In this case the i'th element of $\hat{d}$ can be expressed as

$$[\hat{d}]_i = \max_{d \in D_k^N} [G]_i d, \tag{3}$$

since this leads to the tightest constraints in the inequality (and thus the most conservative restriction on valid combinations of starting points and input sequences). Here $[G]_i$ denotes the i'th row of the matrix G. By using the theory of polyhedral convexity it can furthermore be shown that only the extreme points $\overline{D}_k^N$ of the polytope D_k^N have to be considered in the maximization, which leaves us with a finite number of points to check in order to find $\hat{d}$.

We want to calculate the set S_0 of possible starting points for which a valid control sequence $\hat{u} = u(0), u(1), \ldots u(N-1)$ exists. This leads to the following solution of the reachability problem, which is on the form of a projected polytope.

Proposition 1: Suppose L, M, G and $\hat{d}$ are as above, and that S_0 is the projection of the polytope $\{x \in \mathbb{R}^{n+Nm} \mid Lx \leq M - \hat{d}\}$ onto its first n coordinates, i.e,

$$S_0 = \left\{ s \in \mathbb{R}^n \mid \exists \hat{u} \in \mathbb{R}^{Nm} \text{ s.t. } L \begin{bmatrix} s \\ \hat{u} \end{bmatrix} \leq M - \hat{d} \right\}.$$

Then, the reachability problem is feasible for any $s(0) \in S_0$.

To calculate the initial set S_0, we use an iterative idea that comes from [20]. In the procedure, for which the pseudo code is given in Algorithm 1, the feasible initial set S_0 is computed by back-propagating the final set ς_j one time step at a time.

Algorithm 1 Reachability calculation

1: **syntax**: SOLVE_FEASIBLE(System dynamics $\mathcal{M}_k$, discrete state ς_i, discrete state ς_j, horizon length N, mapping $P_{\mathcal{S}}$)
2: $S_0 \leftarrow \varsigma_j$
3: $i = 0$
4: **while** $i < N$ **do**
5: Build L,M,G and compute $\hat{d}$, using $N = 1$ and S_0 as the final set.
6: $S_0 \leftarrow proj_n(\{x \mid Lx \leq M - G\hat{d}\})$
7: $i \leftarrow i + 1$
8: **end while**
9: **return:** S_0

The advantage of this iterative approach lies in how it deals with disturbances. Instead of calculating the whole sequence of N input signals at one time, we recalculate the input in each time step and can thus take into account additional information, namely all disturbances up to the current time step in the iteration. When implementing a controller, it is accordingly also necessary to recompute the input signal at every time step, after measuring the latest disturbance. For each call to Algorithm 1, N projections of a polytope of dimension $n+m$ are computed.

Remark 1: We assume the mode remains constant in N steps, which is the fixed time horizon of the abstraction. Hence, N should be chosen by taking into account the time scale of the changes in the environment variable e_d, controlling the switches, and can be set to 1 for arbitrarily fast switching. Choosing $N > 1$ requires additional synchronization assumptions (i.e., e_d only changes at times $t = cN$ for integer c). In the context of VMS, environment for a subsystem is typically the other subsystems [17]. Hence, in VMS applications it is possible to achieve synchronization for $N > 1$, either by using synchronous architectures or through communication.

B. Specification Guided State Space Discretization

In order to be able to establish reachability relations between the discrete states, in general the state space has to be partitioned further. We do this at the same time as we look for reachability relations. If it turns out after solving the reachability problem for a pair of cells that the final cell is reachable from only a part of the initial cell, the idea is to divide the initial cell into one feasible and one non-feasible part. This discretization algorithm can be found in Appendix I.

We do additional discretization separately for each dynamical mode $\mathcal{M}_k$ to obtain N_d different partitions of the original continuous state space. Then we merge these partitions into one final single partition $\mathcal{S}$ which will have the reachability characteristics of all N_d partitions. The merging procedure is given in Algorithm 2 and is essentially making new discrete states out of all possible intersections. Finally we recompute the possible transitions for this new partition $\mathcal{S}$ using each dynamical mode. This has to be done since the reachability relations may not be preserved when cells are divided. We denote the set of possible transitions for mode k by the transition relation $R_k \subseteq \mathcal{S} \times \mathcal{S}$. In particular, for $\varsigma_i, \varsigma_j \in \mathcal{S}$, $(\varsigma_i, \varsigma_j) \in R_k$ if ς_j is reachable from ς_i in the sense of Definition 2. The partition $\mathcal{S}$ together with the N_d transition relations R_k, $k \in \{1, \ldots, N_d\}$, is our finite transition system $\mathbb{D}$.

Algorithm 2 Merging

1: **syntax**: MERGE(List of partitions $\{\mathcal{S}_1, \mathcal{S}_2, \ldots \mathcal{S}_{N_d}\}$)
2: $\mathcal{S} \leftarrow \mathcal{S}_1$
3: **for** $k \in \{2, 3, \ldots, N_d\}$ **do**
4: $\mathcal{S}_{temp} \leftarrow \emptyset$
5: **for** $\varsigma_i \in \mathcal{S}$ **do**
6: **for** $\varsigma_j \in \mathcal{S}_k$ **do**
7: **if** $\varsigma_i \cap \varsigma_j \neq \emptyset$ **then**
8: $\mathcal{S}_{temp} \leftarrow \begin{bmatrix} \mathcal{S}_{temp} & \varsigma_i \cap \varsigma_j \end{bmatrix}$
9: **end if**
10: **end for**
11: **end for**
12: $\mathcal{S} \leftarrow \mathcal{S}_{temp}$
13: **end for**
14: **return**: $\mathcal{S}$

C. Planner Synthesis

In this section, we discuss how to synthesize a discrete plan, given the finite transition system $\mathbb{D}$ constructed in the previous section and a temporal logic specification of the assume–guarantee form (2). We first recast the problem as a two player temporal logic game by treating the environment driven switches as adversaries. This reformulation allows us

to employ a method due to Piterman et al. [9], [21] for synthesis of reactive discrete controllers in the presence of adversarial environment. Synthesis of reactive controllers for a general LTL formula has prohibitive complexity. However, as shown in [21], for an expressive subset of LTL, the so-called *Generalized Reactivity(1)* (GR(1)) formulas, synthesis can be achieved in time quadratic in the state space size (i.e., $|\mathcal{V}|$). We restrict ourselves to GR(1) formulas in this paper. Next, we give a very brief overview of the setup in [9], [21] and show how it relates to the problem we consider in this paper.

The main idea is to pose the planner synthesis problem as a two player game between the discrete states $\mathcal{S}$ in $\mathbb{D}$ and the set of environment variables, $\mathcal{E}$, that determine the dynamical mode. Consider the following GR(1) specification:

$$\varphi \doteq (\varphi_e \rightarrow \varphi_s), \tag{4}$$

where for $\alpha \in \{e,s\}$ both φ_α have the following structure :

$$\varphi_\alpha \doteq \theta^\alpha_{init} \wedge \bigwedge_{i\in I_{t\alpha}} \Box\psi^\alpha_i \wedge \bigwedge_{i\in I_{g\alpha}} \Box\Diamond J^\alpha_i.$$

In Eq. (4), φ_e characterizes the assumptions on the environment and φ_s describes the correct behavior of the system. In particular, θ^α_{init} is an atomic proposition characterizing the initial conditions; ψ^α_i are atomic propositions characterizing invariants that should always be satisfied (e.g. safety requirements); and J^α_i are atomic propositions characterizing states that should be attained infinitely often. We assume (4) does not contain the next operator ($\bigcirc$), hence it is stutter invariant. This technical assumption can be relaxed for the case when $N = 1$. From (4) and $\mathbb{D}$ we construct an additional specification that describes all allowable moves:

$$\varphi_d \doteq (\varphi_e \rightarrow \varphi_s \wedge \varphi_{\mathbb{D}}), \tag{5}$$

where $\varphi_{\mathbb{D}}$ encodes the transition system $\mathbb{D}$. In particular, $\varphi_{\mathbb{D}} \doteq \bigwedge_{k=1}^{N_d} \bigwedge_{\varsigma_i\in\mathcal{S}} \Box((e_d = k \wedge s \in \varsigma_i) \rightarrow \bigcirc \bigvee_{(\varsigma_i,\varsigma_j)\in R_k} (s \in \varsigma_j))$. Since, we abstract the discrete model with fixed time horizon N, next operator ($\bigcirc$) in (5), corresponds to N steps later for the continuous dynamics and by construction, the propositions on s remain unchanged during these N steps. Note that if (4) is in GR(1) so is (5). Hence, the synthesis method proposed in [9], which is implemented in JTLV [22], can be used. If there exists a discrete plan that satisfies (5), we say the specification (or the synthesis problem) is *realizable*. In this case, the output of the synthesis algorithm is a partial function $f : (e(t), \varsigma(t), e(t+N)) \mapsto \varsigma(t+N)$ (possibly with some finite memory variable as an additional input argument that we ignore for simplicity of notation), represented by a finite state automaton, that maps the previous discrete system state $v = (\varsigma, e)$ and the current environment value e to the next discrete plant state in the partition. We call f the *planner* for the system.

D. Implementation and Correctness of the System

The states $\mathcal{S}$ in the finite transition system $\mathbb{D}$ is a subpartition of the initial proposition preserving partition $\mathcal{P}$. In this case it is trivial to show that $\mathcal{S}$ also is proposition preserving. Then a discrete planner can be synthesized using JTLV [22] by giving $\mathbb{D}$ and the corresponding set of atomic propositions for each discrete state as input, as described in III-C.

During execution, the planner will control between which discrete states the plant should move in order to fulfill the system specifications. In order to perform these discrete transitions, we need a continuous controller. We do this with finite-horizon linear quadratic regulator (LQR), using the (convex) constraints of reachability in III-A. Like with the reachability algorithm, we must propagate the final set backwards in time to find the feasible set for each time step. In order to move from ς_i to ς_j we calculate an input sequence u^* by solving the following problem for a given initial position $s(0) \in T_{\mathcal{S}}^{-1}(\varsigma_i)$,

$$u^* = \begin{cases} \arg\min_{\{u(t)\}_0^{N-1}} & \sum_{t=1}^{N}(s(t)^T Q_t s(t) + q_t^T s(t)) + \\ & \sum_{t=0}^{N-1}(u(t)^T R_t u(t)), \\ s.t & s(N) \in T_{\mathcal{S}}^{-1}(\varsigma_j), \\ & s(1) \in S_{0_{N-1}}, \\ & s(t) \in T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i)), \\ & u(t) \in U(s(t)), \\ & \forall t \in \{1,2,\ldots,N-1\}, \end{cases} \tag{6}$$

where $S_{0_{N-1}} \subseteq T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$ is the set calculated by Algorithm 1 for which the reachability problem is feasible in $N-1$ steps. By allowing the plant to move inside the whole original proposition preserving cell $T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$ during the transition, rather than enforcing it to stay inside $T_{\mathcal{S}}^{-1}(\varsigma_i)$ like has been done previously, we optimize over a larger set of trajectories. Therefore the cost of the optimal trajectory for this relaxed problem is at least as low as before, and often lower.

Only the first input signal $u(0)$ is used at each time step. When the plant has moved one step the new disturbance is taken into account to compute a new input signal, again using (6) but with the horizon length N decreased by 1.

By enforcing the trajectory of a transition to stay inside the proposition preserving cell, the following statement can be made about the system behavior.

Proposition 2: Under the assumption that the switches only occur at times $t = cN$ for integer c, using the planner f synthesized for (5) together with the controller (6) guarantees that the specification φ in (4) is satisfied by the system.

Proof: The result follows from Proposition 1 in [6] together with stutter invariance assumption on φ and the fact that we require that the trajectory of the transitions between two discrete states should lie inside the parent cell $T_{\mathcal{P}}^{-1}(P_{\mathcal{S}}(\varsigma_i))$ in the original proposition preserving partition. Hence, the same set of propositions are guaranteed to be satisfied during the transitions. Therefore the planner/controller system will behave correctly with respect to the specification φ in (4). ■

E. Comparison with Earlier Work

In addition to extending the temporal logic planning framework to switched systems, in this paper we make a better use of proposition preserving partitions. In particular,

we use a less restrictive definition of reachability than what has been used in earlier papers that employ the discretization algorithm given in Appendix I. Instead of enforcing the plant to remain inside the initial cell, i.e $s(t) \in T_S^{-1}(\varsigma_j)$, during a discrete transition, we use the looser constraint $s(t) \in T_{\mathcal{P}}^{-1}(P_S(\varsigma_j))$. A looser definition of reachability makes it possible to establish more reachability relations, something that is crucial in order making the controller synthesis realizable.

To illustrate the advantage of the new approach, we compare the two methods on a simple two-dimensional example with the following (not switched) dynamics:

$$\begin{bmatrix} s_1(t+1) \\ s_2(t+1) \end{bmatrix} = \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} + \begin{bmatrix} d_1(t) \\ d_2(t) \end{bmatrix}, \quad |u_1(t)+u_2(t)| \leq 0.16, \quad |d_1(t)|, |d_2(t)| \leq 0.04. \tag{7}$$

The state space is the set $\{x, y \in \mathbb{R}^2 \mid 0 \leq x \leq 3, 0 \leq y \leq 2\}$ and the initial proposition preserving partition consists of 6 identical square cells. We discretize the state space with the algorithm in Appendix I, both using the notation of reachability in Definition 2 and the corresponding definition from [7]. The specifications on the system require the plant to be able to move between the lower left and the upper right cells in the initial partition. It turns out that with this set-up, discretizing using the reachability constraints in [7] results in 29 possible transitions (including self transitions) between a total of 23 discrete states, which in this case is not enough to make the specification realizable. However, using the new relaxed constraints, a total of 34 discrete states are created with 93 transitions between them, which makes the controller synthesis realizable.

IV. EXAMPLE

The presented strategy for automatic controller synthesis is implemented on a fuel tank system. To build the controller we use The Temporal Logic Planning (TuLiP) Toolbox [23], a software package that is designed to synthesize controllers using the presented method, including an interface to JTLV.

The dynamical system we look at consists of two fuel tanks T_1 and T_2 in an airplane and the state variables are the two fuel volumes v_1 and v_2 in these tanks. We suppose that the maximum capacity in both tanks is 10 fuel units. The system can be controlled by moving fuel from tank T_1 to tank T_2 by using a pump, at a maximum rate of 3 fuel units per time step. This problem has only one control signal but two state variables, so it is not fully actuated. Such systems are often hard to control.

We model the system with two dynamical modes, one for *normal* operation and one for *aerial refueling* mode. During aerial refueling mode a tank plane is flying next to our plane and fills up fuel in tank T_1 at a rate of 3 fuel units per time step. We assume that the plane consumes 1 fuel unit per time step and that this fuel is taken from tank T_2. This gives the

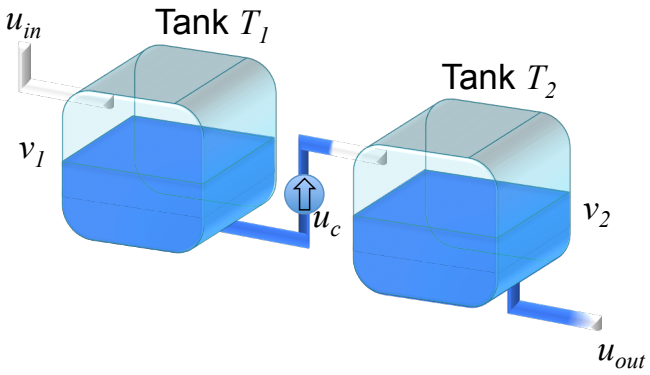

Fig. 1. Fuel tank system.

following model for the dynamics:

$$\begin{bmatrix} v_1(t+1) \\ v_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} u_c(t) + \begin{bmatrix} u_{in}(t) \\ -1 \end{bmatrix}, \tag{8}$$

where $u_{in}(t)$ can be either 0 or 3, depending on the dynamical mode. The restriction on the input signal can be written on polytope form as

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} u_c(t) \\ v_1(t) \end{bmatrix} \leq \begin{bmatrix} 3 \\ 0 \\ u_{in}(t) \end{bmatrix}, \tag{9}$$

where the last inequality makes sure that not more fuel than what is available can be moved from tank T_1.

A. Specifications

What we want to achieve with this system is to keep the difference in fuel level low between the two tanks. The motivation for this is that a big difference may cause instability in the airplane. It is assumed that the refueling mode will be initiated when the levels are too low, so that the system will never run out of fuel completely, and that the refueling mode will stop when tank T_2 is almost full. We therefore give the following requirements and specifications on the system.

Assumptions:

- When $v_1 + v_2 \leq 2$ and $u_{in} = 0$, next step require $u_{in} = 3$.
- When $v_2 \geq 8$ and $u_{in} = 3$, next step require $u_{in} = 0$.

Requirements:

- Always require that $|v_1 - v_2| \leq 2$.
- Always eventually require that $|v_1 - v_2| \leq 1$.

The first requirement ensures safe operation, while the second requirement makes sure that improved performance is achieved infinitely often.

B. State Space Discretization

We run the discretization algorithm in Appendix I two times to obtain one partition for each dynamical mode, using time step $N = 1$ for reachability in one time step (allowing for the system to switch between dynamical modes at every time step). Then we merge the two partitions using Algorithm 2, the resulting final partition of the state space is shown in

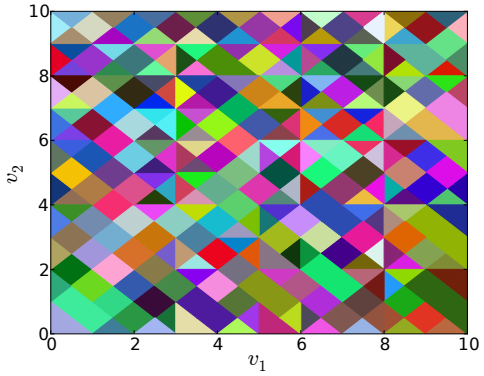

Fig. 2. Final state space partition, each colored area represents a discrete state.

Fig. 2. Finally we make calls to Algorithm 1 in order to get two lists of possible transitions between discrete states in this partition, one list for each dynamical mode. The partition and the possible transitions can now be plugged into JTLV to synthesize a discrete planner, and it turns out that the specifications are realizable for this particular problem. The discrete states that fulfill the specification are the states that lie at most 2 length units away from the line $v_1 = v_2$. Since the planner synthesis is successful, we have a way of controlling the system to always stay inside this area.

For this example, the discretization procedure took 282 seconds on a MacBook Air 1.86 GHz. A planner for the resulting 277 discrete states could be synthesized in 15 seconds on the same computer.

C. Results

The planner is implemented in a simulation together with a controller on the form of (6). We put a weight in the optimization that punishes deviations from the centers of the cells. The simulation is run in 28 time steps and the result is shown in Fig. 3. As can be seen, the two levels of fuel track each other closely, fulfilling the system specification. In the beginning of the experiment the plane is in normal operation mode, which can be seen by the declining fuel levels. When the fuel levels become low the system is switched to aerial refueling mode and fuel starts to be added to tank T_1, causing both levels to increase. Fig. 4 shows how much fuel that is being moved from tank T_1 to tank T_2 during the simulation.

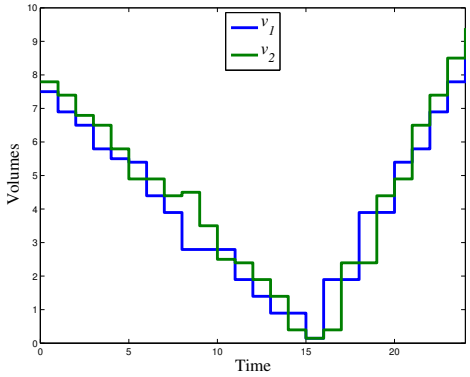

Fig. 3. Tank levels in simulation.

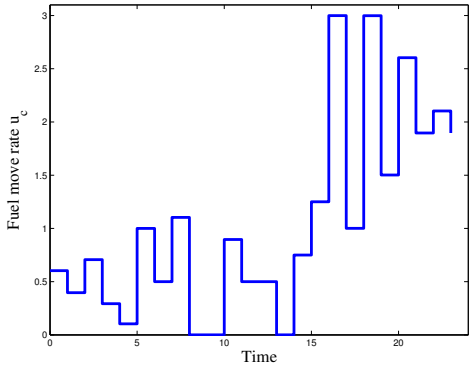

Fig. 4. Fuel move rate, $u_c(t)$, in simulation.

V. CONCLUSIONS AND FUTURE WORKS

We have presented a strategy for automatic synthesis of a planner and a controller for a switched system that is correct by construction, in the sense that once synthesized, the planner/controller is guaranteed to fulfill the specifications. The strategy can be implemented on systems that are switched, i.e. whose dynamical mode is controlled by the environment. Most of the results on switched systems are either only sufficient conditions or computationally expensive; and the proposed method is no exception. However, as shown in the paper, controller synthesis for switched systems from temporal logic specifications, can be recast as a two-player game by treating the switching sequence as environment. This reformulation has enabled using tools from reactive controller synthesis for linear or piecewise affine systems with appropriate modifications. An important step in the procedure is to partition the continuous state space into a finite number of discrete states, in order to enable planning. We have shown improved performance in discretization step by effectively using propositions in the temporal logic formula while computing reachability relations.

The planner/controller synthesis strategy has been implemented on an example with fuel tanks. The fuel tank system is under-actuated and is switched by the environment between two dynamical modes. A planner could successfully be synthesized and therefore any execution is guaranteed to fulfill the specification.

Future work includes tuning the discretization procedure for switched systems by taking into account the a priori information on the switches to reduce potential conservatism. We are also interested in investigating the convergence properties of the approach in the sense that as finer and finer partitions are considered whether it would be possible to obtain a control strategy, that satisfies the specification, at discrete level whenever one exists for the continuous system.

APPENDIX I
DISCRETIZATION ALGORITHM

This appendix summarizes a discretization algorithm proposed in [6], [7], [19] and also used in this paper. The purpose of this algorithm is to refine a given partition of the state space and at the same time establish reachability relations between discrete states in the partition.

An iteration of the proposed algorithm starts with picking two arbitrary states ς_i and ς_j in the current partition. Then the subset $S_0 \subseteq \varsigma_i$ from which ς_j is reachable is computed. If S_0 is found to be non-empty, but not the same as ς_i, ς_i is partitioned into $\varsigma_i \cap S_0$ and $\varsigma_i \backslash S_0$, and thus the number of discrete states increases by 1. By construction the state ς_j will be reachable from the new state $\varsigma_i \cap S_0$. The idea is to keep doing this until enough transitions have been established inside the state space. The pseudo-code of this algorithm is given in Algorithm 3, where a minimum cell volume is used as a termination criteria. To improve the speed of the algorithm, it can be restricted it to check for transitions between states that are neighbors of some order N_{NB}, using the notation introduced in II-A.

Algorithm 3 State space discretization

syntax: DISCRETIZE(Proposition preserving partition $\mathcal{P}$, dynamic mode $\mathcal{M}_k$, neighbor order N_{NB}, horizon length N, minimum volume *min_vol*)
Initialize transition matrix $TR \leftarrow zeros(n \times n)$.
Initialize new partition $\mathcal{S} \leftarrow copy(\mathcal{P})$.
Build a neighbor matrix NB from $\mathcal{S}$ s.t $NB(i,j) = 1$ if ς_i and ς_j are neighbors.
Build matrix of transitions to check $IJ \leftarrow (NB^{N_{NB}} > 0)$.
while $sum(IJ) > 0$ **do**
  Pick (i,j) such that $IJ(j,i) = 1$, then set $IJ(j,i) = 0$.
  $S_0 \leftarrow$ SOLVE_FEASIBLE($\mathcal{M}_k$, ς_i, ς_j, N, P)
  if (vol($\varsigma_i \cap S_0$) $>$ *min_vol*) AND (vol($\varsigma_i \backslash S_0$) $>$ *min_vol*) **then**
    $\mathcal{S}(i) \leftarrow \varsigma_i \cap S_0$
    $\mathcal{S} \leftarrow \begin{bmatrix} \mathcal{S} & \varsigma_i \backslash S_0 \end{bmatrix}$
    Add row and column of zeros to TR, NB and IJ.
    Update $NB(end,:)$, $NB(:,end)$, $NB(i,:)$, $NB(:,i)$
    $TR(:,end) \leftarrow TR(:,i)$
    if $i \neq j$ **then**
      $TR(j,i) \leftarrow 1$
    end if
    #Update IJ matrix with transitions to check:
    $IJ(i,:) \leftarrow (NB^{N_{NB}}(i,:) - TR(i,:) > 0)$
    $IJ(:,i) \leftarrow (NB^{N_{NB}}(:,i) - TR(:,i) > 0)$
    $IJ(end,:) \leftarrow (NB^{N_{NB}}(end,:) - TR(end,:) > 0)$
    $IJ(:,end) \leftarrow (NB^{N_{NB}}(:,end) - TR(:,end) > 0)$
  else if vol($\varsigma_i \backslash S_0$) $= 0$ **then**
    $TR(j,i) \leftarrow 1$
  end if
end while
return: New partition $\mathcal{S}$, transition matrix TR.

REFERENCES

[1] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.
[2] A. Pnueli, "Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends," *Current Trends in Concurrency. Overviews and Tutorials*, pp. 510–584, 1986.
[3] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.
[4] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.
[5] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.
[6] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Proc. IEEE Conference on Decision and Control (CDC'09)*, Shanghai, China, Dec. 2009.
[7] ——, "Automatic synthesis of robust embedded control software," in *AAAI Spring Symposium on Embedded Reasoning: Intelligence in Embedded Systems*, 2010, pp. 104–111.
[8] ——, "Receding horizon control for temporal logic specifications," in *HSCC*, 2010, pp. 101–110.
[9] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *VMCAI*, 2006, pp. 364–380.
[10] D. Liberzon, *Switching in systems and control*, ser. Systems & control. Birkhäuser, 2003.
[11] M. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 4, pp. 475 –482, apr 1998.
[12] J. P. Hespanha, "Uniform stability of switched linear systems: Extensions of LaSalle's invariance principle," *IEEE Trans. on Automat. Contr.*, vol. 49, no. 4, pp. 470–482, Apr. 2004.
[13] P. A. Parrilo and A. Jadbabaie, "Approximation of the joint spectral radius using sum of squares," *Linear Algebra and its Applications*, vol. 428, no. 10, pp. 2385 – 2402, 2008, special Issue on the Joint Spectral Radius: Theory, Methods and Applications.
[14] F. Blanchini, S. Miani, and F. Mesquine, "A separation principle for linear switching systems and parametrization of all stabilizing controllers," *Automatic Control, IEEE Transactions on*, vol. 54, no. 2, pp. 279 –292, feb. 2009.
[15] D. Tarraf, A. Megretski, and M. Dahleh, "Finite approximations of switched homogeneous systems for controller synthesis," *Automatic Control, IEEE Transactions on*, vol. 56, no. 5, pp. 1140 –1145, may 2011.
[16] I. Moir and A. Seabridge, *Aircraft Systems: Mechanical, Electrical, and Avionics Subsystems Integration*. AIAA Education Series, 2001.
[17] N. Ozay, U. Topcu, and R. M. Murray, "Distributed power allocation for vehicle management systems," in *Proc. IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, USA, Dec. 2011, (to appear).
[18] P. Derler, E. Lee, and S.-V. A., "Addressing modeling challenges in cyber-physical systems," EECS Department University of California, Berkeley, Tech. Rep., March 2011. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-17.pdf
[19] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," 2010, submitted to IEEE Transactions on Automatic Control.
[20] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*. Springer, Lecture Notes in Control and Information Sciences, vol 290, 2003.
[21] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," *Journal of Computer and System Sciences*, 2011, (in print). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000011000869
[22] A. Pnueli, Y. Sa'ar, and L. D. Zuck. (2010) JTLV : A framework for developing verification algorithms. 22nd International Conference on Computer Aided Verification. [Online]. Available: http://jtlv.ysaar.net
[23] T. Wongpiromsarn, U. Topcu, N. Özay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *HSCC*, 2011, software available at http://tulip-control.sf.net/.